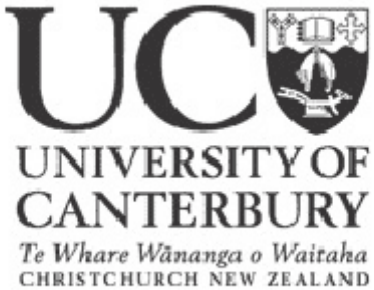
Department of Physics and Astronomy, University of Canterbury,
Private Bag 4800, Christchurch, New Zealand

A 3D Computer Vision System in Radiotherapy Patient Setup

Te-yu Chyou

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at the University of Canterbury



Supervisor: Dr. Juergen Meyer

Abstract

An approach to quantitatively determine patient surface contours as part of an augmented reality (AR) system for patient position and posture correction was developed.

Quantitative evaluation of the accuracy of patient positioning and posture correction requires the knowledge of coordinates of the patient contour. The system developed uses the surface contours from the planning CT data as the reference surface coordinates. The corresponding reference point cloud is displayed on screen to enable AR assisted patient positioning. A 3D computer vision system using structured light then captures the current 3D surface of the patient. The offset between the acquired surface and the reference surface, representing the desired patient position, is the alignment error.

Two codification strategies, spatial encoding, and temporal encoding, were examined. Spatial encoding methods require a single static pattern to work, thus enabling dynamic scenes to be captured. Temporal encoding methods require a set of patterns to be successively projected onto the object, the encoding for each pixel is only complete when the entire series of patterns has been projected.

The system was tested on a camera tracking object. The structured light reconstruction was accurate to within ±1 mm, ±1.5 mm, and ±4 mm in *x*, *y*, and *z*-directions (camera optical axis) respectively. The method was integrated into a simplified AR system and a visualization scheme based on *z*-direction offset was developed. A demonstration of how the final AR-3D vision hybrid system can be used in a clinical situation was given using an anatomical teaching phantom.

The system and visualisation worked well and demonstrated the proof of principal of the approach. It was found that the achieved accuracy was not yet sufficient for clinical use. Further work on improving the projector calibration accuracy is required. Both the camera registration process and 3D computer vision using structured light have been shown to be capable of sub-millimeter accuracy on their own. If that level of accuracy can be reproduced in this system, the concept presented can potentially be used in Oncology departments as a cost-effective patient setup guidance system

for external beam radiotherapy, used in addition to current laser/portal imaging/cone beam CT based setup procedures.

Acknowledgement

Firstly, a big thank you to the supervisor of my project, Dr. Juergen Meyer, for his support, guidance, and patience throughout the project. Special thanks also to Dr. Adrian Clark of HITLab NZ, who provided me with the cameras, programming camera drivers, and offered his expertise regarding the Kinect. Finally, I would like to thank the rest of the Medical Physics group in room 813, Syen, Alicia, James, Qing, Naomi, and Steve for giving me such a friendly and supportive environment to work in.

Table of Content

Chapter 1

Introduction

1.1 Background

In clincal oncology, there are three principal modalities for treating malignant tumours: surgery, chemotherapy, and radiotherapy [1, 2]. External beam radiotherapy is a non-invasive procedure that uses ionizing radiation to produce biological damage of cells in the target volume. Cells are damaged by radiation either directly through Coulomb interactions with critical structures such as the DNA, or indirectly through the diffusion of radiation induced free radicals, into critical structures of the cells [1, 2].

The outcome of treatment depends on the response of the tumour cells to ionizing radiation. Ideally, tumour cells should receive irreversible damage that leads to reproductive failure and cell death, thus halting proliferation of the malignant growth [2]. The effectiveness of ionizing radiation in achieving this may be represented by means of the tumour control probability (TCP). In general, TCP increases with dose, however higher radiation doses also lead to more damage to healthy tissues, particularly around the proximity of the tumour. The impact of ionizing radiation on healthy tissue may be expressed by the normal tissue complication probability (NTCP).

The goal of radiotherapy treatment planning is to achieve the highest possible TCP while maintaining NTCP below approximately 5%. The ratio of TCP to NTCP is called the therapeutic ratio, typically in radiotherapy, this should be as high as possible.

In treatment planning, a three dimensional image of the patient is obtained through a computed tomography (CT) scan. From the CT data, the treatment target, patient's skin contour, and other organs of interest are identified [1]. The planner or an optimization program uses these parameters to calculate the theoretical beam shapes and orientations that would deliver the prescribed dose to the planning target volume while minimizing doses to the skin and other organs of interest.

The standard method for patient alignment on the treatment couch is through use of skin marks placed at the time of treatment simulation and treatment planning. The markers indicate the location of the treatment isocenter. During patient setup the same isocenter position can be reproduced by aligning each skin mark to the appropriate room laser [1, 2].

A significant contributing factor to radiotherapy incidents has been patient positioning related errors, the most common being inaccurate alignment whereby geographical misses of the target area by several centimeters resulted in unintended doses to healthy tissue [4, 5, 6]. Misalignment of the patient on a bigger magnitude, for example left/right and anterior/posterior confusions have also been reported [6].

In treatment, radiation is typically delivered in fractions for radiobiological reasons to improve the therapeutic ratio [3]. One source of error in this process is the discrepancies between the patient position and postures during the CT scan and treatment planning, and those during the actual treatment sessions. Since a treatment can consists of 30 or more fractions, daily patient setup errors can add up and decrease the overall quality of the treatment. At present, on-line MV and kV x-ray imaging is a widely accepted practice for patient setup in radiotherapy.

Electronic portal imaging devices (EPID) use either a metal plate-phosphor screen combination, or a matrix of liquid filled ionization chambers along the linac beam line to produce beam's eye views (BEV) [1]. Portal images can be compared to digitally reconstructed radiographs (DRRs) from the treatment planning system and treatment simulation to verify the position of the treatment isocenter.

Gantry-mounted kV imaging systems are becoming a key feature in modern medical accelerator design [7]. Unlike MV EPID, gantry-mounted kV systems such as Elekta's Synergy XVI and Varian's On-Board Imager (OBI) use a separate kV x-ray tube mounted on the gantry, orthogonal to the linac treatment head and any EPI systems associated with it. Gantry-mounted kV imaging systems can perform radiography, fluoroscopy and most importantly, the in-room cone beam computed tomography for volumetric assessment.

Image guided radiotherapy (IGRT) utilizing cone beam computed tomography (CBCT) is widely practiced. IGRT systems allow pretreatment imaging of the patient on the linac treatment table. The use of CBCT imaging enables visualization of the exact tumour location, as well as volumetric data of patient anatomy. The CBCT data are

compared to planning CT data and the treatment couch is translated and rotated to achieve the desired position in a procedure known as rigid-body transformation. One of the shortcomings of the rigid-body transformation based setup approach is that patient deformations, or changes in posture, due to incorrect positioning or changes in the anatomy due to swelling or weight loss cannot be easily detected and corrected.

A variety of photogrammetric systems have been proposed and tested for the purpose of detecting and reducing planning-to-setup, and treatment-to-treatment positioning errors. The majority of existing systems is either stereovision based [8, 9, 10, 11, 12, 13], or utilizes on-patient markers [14].

In rigid-body transformation, the entire patient is shifted to align the target volume, without posture adjustment. However, of the human anatomy, only bony structures and structures rigidly related to bone can be accurately modeled as rigid bodies. For other parts such as the prostate and the breast, position of the organ and the state of the surrounding anatomy are affected by breathing motion [15, 16] and posture [17]. Furthermore, the typical patient setup procedure involves the use of skin marks to assist day-to-day setup [1, 18], however, the skin itself is not a rigid structure and can vary through the course of the treatment as the result of obesity, variations in muscle tone, or patient posture [18]. Rigid-body transformation should ideally be used with posture corrections to minimize the geometric offset from geometry used in treatment planning.

1.2. Existing Augmented Reality System

Augmented Reality (AR) has been proposed by our group at the University of Canterbury and investigated as a means to provide visual guidance during patient set up [19]. AR is a form of computer virtual reality in which real world elements captured by camera are enhanced by the addition of computer generated graphics. A brief overview of the existing AR system [19] is given here.

The surface contour of the patient is obtained from the patient's pre-treatment CT scan, which is already the standard practice in 3D external beam radiotherapy. The contour data is available as a DICOM-RS object. DICOM (Digital Imaging and Communications in Medicine) is the standard file format for storing and transferring images between medical equipment. A DICOM-RS object, where RS stand for "Radiotherapy Structure", refers to a DICOM object containing contours for volume

definitions in radiotherapy. The volumes of interest usually include several levels of treatment target volumes (e.g. GTV, CTV, PTV) defined by the clinician, the critical structures, as well as the body outline of the patient. A Matlab script is used to specifically extract the surface contour from the DICOM-RS object. The contours are then rendered into a surface using *Meshlab* [20], a 3D surface rendering program.

A registration cube (see Figure 1.1), similar to those used in the routine quality assurance (QA) of CT alignment lasers, is used to register the camera location and orientation with respect to the linac isocenter. The registration cube is positioned on the treatment couch, with the aid of the room laser, such that its center coincides with the linac isocenter. A single view of the registration cube by a calibrated camera (see section 2.3) is sufficient for the required transformation parameters to be determined. The transformation parameters can then be used to display the 3D model of the patient outline on top of the live camera view of the linac. The cube can be removed once the registration process is done, and patient set-up may begin. During the alignment of the patient, the radiation therapists now have the option to refer to the augmented monitor with the correctly positioned contour on-screen (see Figure 1.2).

Figure 1.1. Registration cube lined up to the room laser [19].

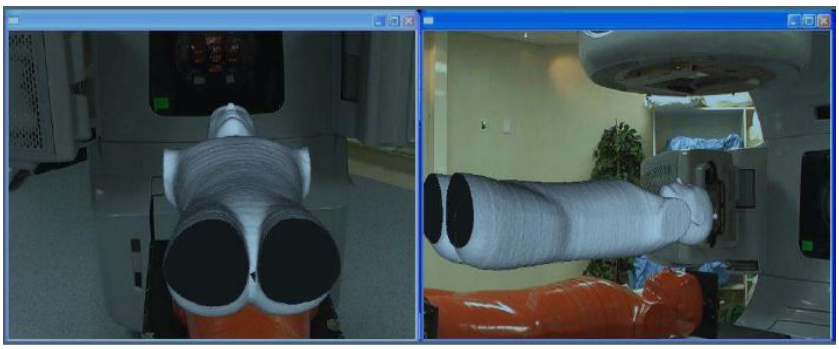


Figure 1.2. Augmented monitor with the correctly positioned contour on-screen in the existing AR system [19].

The accuracy of the AR system in a clinical environment was evaluated experimentally using an anthropomorphic phantom. The phantom was positioned on the linac treatment couch based on the on-screen contour. A CBCT scan of the phantom in its final position was taken, and compared to the CT volume. From the measured offsets between the CT and CBCT volume, a translational error of 3mm, and rotational errors of 0.19^{o}, 0.06^{o} and 0.27^{o} in pitch, roll and yaw respectively, were deduced.

While AR can provide a visualization of optimal patient surface contour for posture adjustment and aid in rigid-body transformation, it remains a visualization tool and provides no quantitative information. To quantify offsets from the optimal contours caused by posture related deformations, recovery of the actual patient contours is necessary so comparison to planning CT data can be made.

1.3. Surface measurement systems

In external beam radiotherapy, high precision in target positioning relative to the treatment planning coordinate system is imperative to the success of treatment. Patient surface tracking has received growing attention in recent years and commercial products specifically designed for radiotherapy patient monitoring have been developed. The most well known is the stereovision based system, AlignRT [21] by VisionRT.

Bert et al (2005) [11] studied AlignRT's capability in detecting patient shift. In their work on patient realignment, a mannequin phantom was set up on the treatment couch, the couch was then purposely misaligned with the new coordinates determined by a random number generator. For each couch position, they used

AlignRT to calculate the transformation required to realign and compared the result to the known transformation. In 54 sets of random couch coordinates, the biggest translational error was 0.75 mm and the biggest rotational error was under 0.1^{o}. Given that the roof mounted pods were 2.7 m away from the phantom and the resolution of the cameras being 1024 by 768 pixels, the hardware could not have achieved sub millimeter spatial resolution (due to the likely size of the field of view, and the relatively small number of available camera pixels), thus demonstrating the importance of optimization software.

AlignRT has since been tested in cranial stereotactic radiotherapy (SRT) and radiosurgery (SRS) [8, 10], which require some of the most accurate and precise patient alignment amongst the present external beam treatment techniques. In one of the studies, AlignRT produced alignment accuracy comparable to current image/marker-based systems [8].

Computer stereovision relies on two or more distinct, two dimensional digital images of the same object to extract three dimensional information. The simplest form of stereovision is the binocular stereo. Binocular stereo uses two identical cameras positioned such that the image planes of cameras are parallel to each other, and the camera centers are at the same height. Recovering depth information involves two key steps, solving the correspondence problem, and calculating the depth from the disparity [22, 23, 24]. For a single point on the object, disparity refers to the difference in position of this point on the respective image plane of each camera. Disparity is inversely proportional to depth and enables the computation of a 3D representation of the object.

One of the main challenges in building a stereovision based 3D surface imaging system is to solve the correspondence problem. Correspondence problem arises because each point on the image seen by one camera has a line of possible choices (the epipolar line) for the corresponding point as seen by the other camera [22, 23, 24]. Correspondence problem is solved by sliding a window along the epipolar line. Correspondence is where pixels within the corresponding windows have similar intensity. Solving the correspondence problem directly can be difficult when the object of interest is the human body. The skin is in most cases featureless and the matching method is therefore susceptible to false matches.

In structured light based techniques, coded patterns are projected onto the scene and the illuminated scene is then viewed from one or more viewpoints [25, 26, 27].

Because the illumination is coded, correspondence between image points and points on the projected pattern can be identified. The process of finding correspondences is known as decoding of the structured light [25, 27], once the captured image points are decoded, the 3D information of these points can be established through triangulation [25].

1.4. Project Aim and Outline of the Approach

The aim of this project is to implement a structured light based 3D surface measuring technique, then customize and integrate the 3D vision technique into the existing AR approach. The final setup guidance system uses AR to provide a visualization of the optimal patient position and posture for initial positioning. The accuracy of the positioning is then determined quantitatively using the 3D surface measurements (refer to Figure 1.3). The ideal system should provide quantitative information on misalignments on a region-to-region basis, instead of the *global* correctness of the alignment. This feature is important, it provides the user with the location and the magnitude of a misalignment caused by a *localized* deformation.

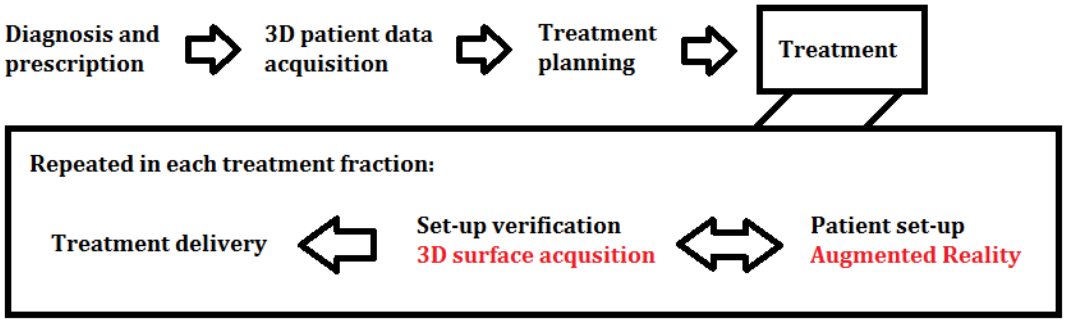


Figure 1.3. Roles of the AR and 3D vision system in the workflow of a typical external-beam radiotherapy treatment course.

The first objective of the project was the calibration of the camera and projector to be used (see section 2.1-2.4). The two main structured light techniques (see section 2.5-2.8) differed only in the way the scene was encoded and decoded, thus they shared the same calibration parameters. After a qualitative comparison of the two encoding techniques, it was found that the temporal encoding method provided better resolution and was less susceptible to decoding errors. In Chapter 3, the accuracy of the structured-light via temporal encoding method was examined quantitatively. Finally, the 3D method was integrated to a simplified AR system (Chapter 4). Figure 1.4 outlines the experimental approach.

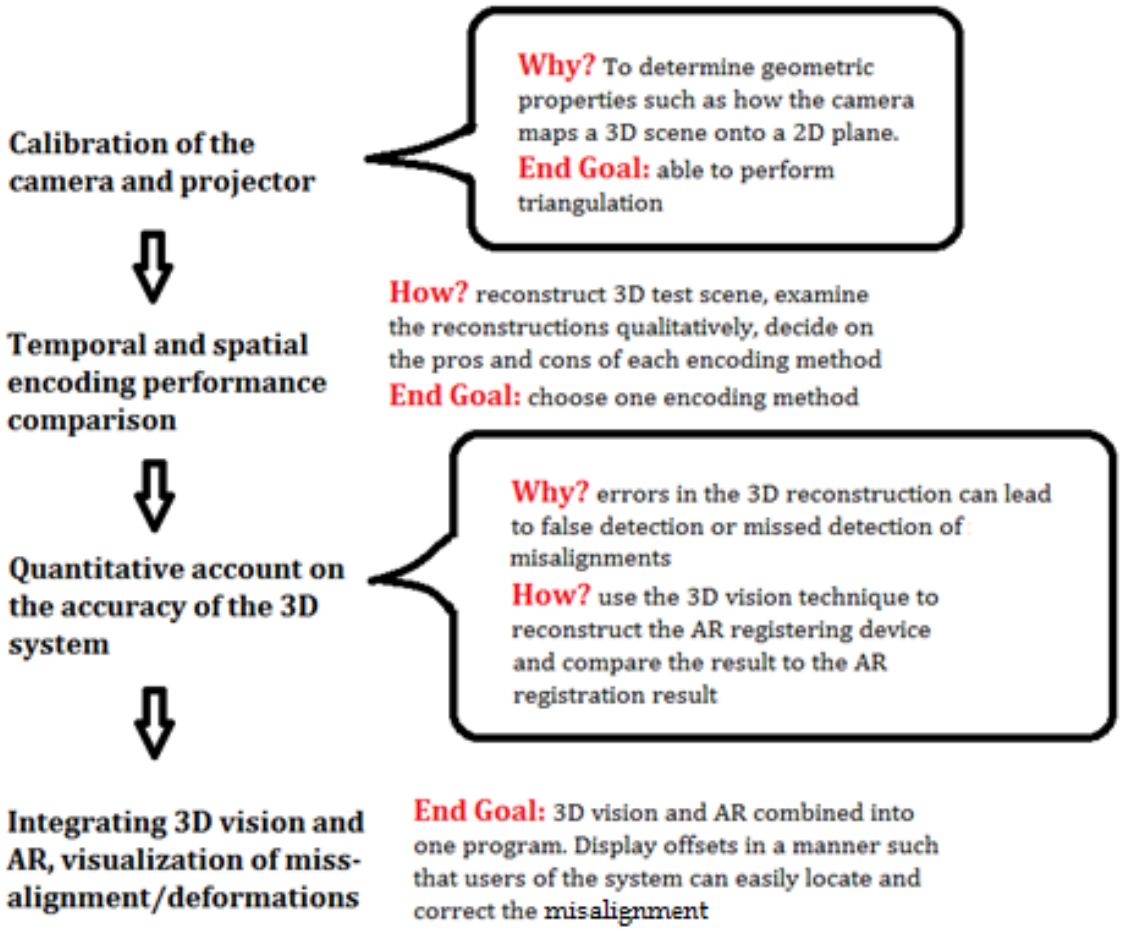


Figure 1.4. Project outline, steps, justifications, and goals.

Chapter 2

Structured Light: Theory and Method

2.1. Triangulation

In a homogenous medium such as air, light traverses in straight lines. Every pixel on a camera image defines a camera ray. A camera ray describes the direction in which a light ray must be incident upon for it to be registered by that camera pixel. It is represented geometrically as a straight line (see equation 1) in 3D space that passes through: 1. the focal point of the camera; 2. the location in the scene from which the light originated. Let **p** be the coordinate of a point in the scene, the vector equation of a camera ray crossing this point takes the form of a straight line:

$$\mathbf{p} = u\mathbf{v} + \mathbf{v}_0 \qquad (2.1)$$

where **v** is the direction vector and $\mathbf{v_0}$ is a point on the line and *u* is a scalar.

A projector works like the inverse of a camera. A dot projected onto the scene forms a projector ray between the focal point of the projector and the point of intersection with objects in the scene. Similarly, a line projected onto the scene, forms a 3D projector plane that contains the focal point and all points on the line segment illuminated by the projection (see Figure 2.1). The equation for a projector plane is:

$$(\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{n} = 0 \qquad (2.2)$$

where **n** is the normal to the plane and $\mathbf{p_0}$ is a point on the plane.

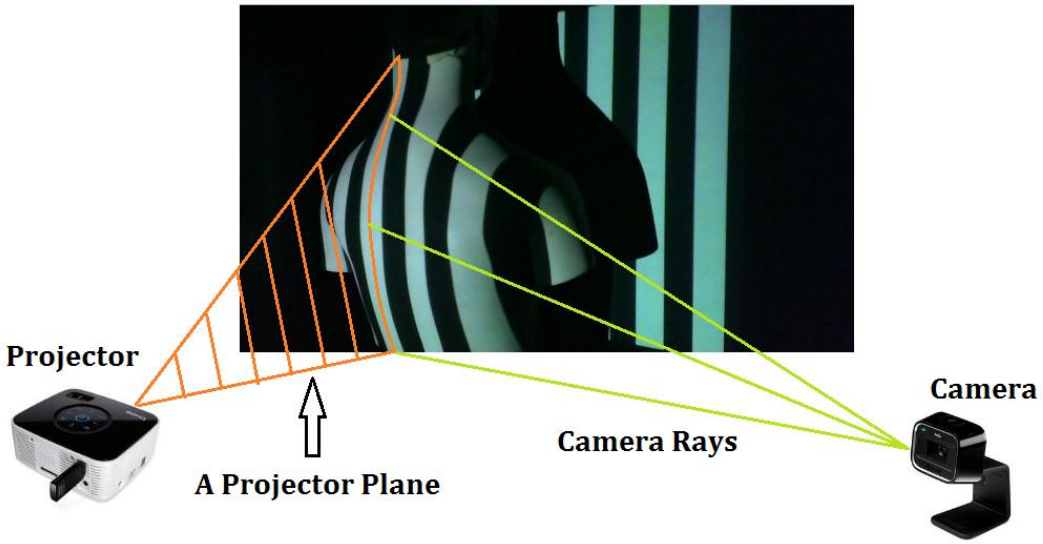


Figure 2.1. Relationship between a projector plane and camera rays. Each stripe pattern projected onto the scene can be thought of as a 3D plane intersecting with object in the scene. When a camera sees the illuminated stripe, the position of each of the illuminated pixels on the image plane relative to the focal point defines the camera rays which form the camera image of the stripe.

The orientation and location of the projector in camera coordinates can be found through calibration. Working in camera coordinates where the camera focal point is the origin, take the location of the camera focal point to be $\mathbf{v_0}$, and let $\mathbf{p_0}$ be the location of the projector. Given an illuminated camera pixel, the distance *u* along the camera ray where intersection with the projector plane occurs can be calculated using equation 2.3, the 3D coordinates of the illuminated point in the scene is found by substituting *u* back to equation 2.1

$$u = \frac{(\mathbf{p}_0 - \mathbf{v}_0) \cdot \mathbf{n}}{\mathbf{v} \cdot \mathbf{n}} \qquad (2.3)$$

2.2. Codification Techniques

For triangulation to work, each camera ray must be intersected with the correct projector plane. To achieve this, the projector must illuminate each part of the scene differently to make regions in the scene distinguishable on the camera images. This step is known as encoding.

Encoding patterns can be divided into two main categories [25]. The temporal encoding techniques, which are characterized by the use of a sequence of encoding patterns, and the spatial encoding techniques that use a single pattern to encode the scene.

In temporal encoding, each part of the scene is given a unique codeword in the form of a colour sequence. The encoding is done by projecting a sequence of patterns onto the scene using a data projector or similar illuminating devices. Since only one pattern can be projected at any instance, the minimum number of camera frames required to capture the whole sequence is the number of encoding patterns. This feature gives the method its name of *temporal* encoding. The maximum resolution of the system is constrained by both the camera and the projector resolution. The maximum number of codewords that can be generated by a data projector is the number of projector pixels. In the situation when the camera used has a much better resolution, a group of neighbouring pixels on the camera captured sequence may share the same codeword, these groups of neighbouring pixels will be referred to as *encoding regions* for the remainder of the thesis.

The specific type of temporal encoding used in this project is called the Gray code structured light sequence. A Gray code sequence is a specific type of binary sequence, obtained from a manner of reflections of a binary sequence (see Figure 2.2). The key feature of a Gray code sequence is that the codeword for two neighbouring encoding region always differ by exactly one bit (see Figure 2.2, bottom) [25], this makes a Gray code sequence more robust to decoding errors than the original binary sequence [25].

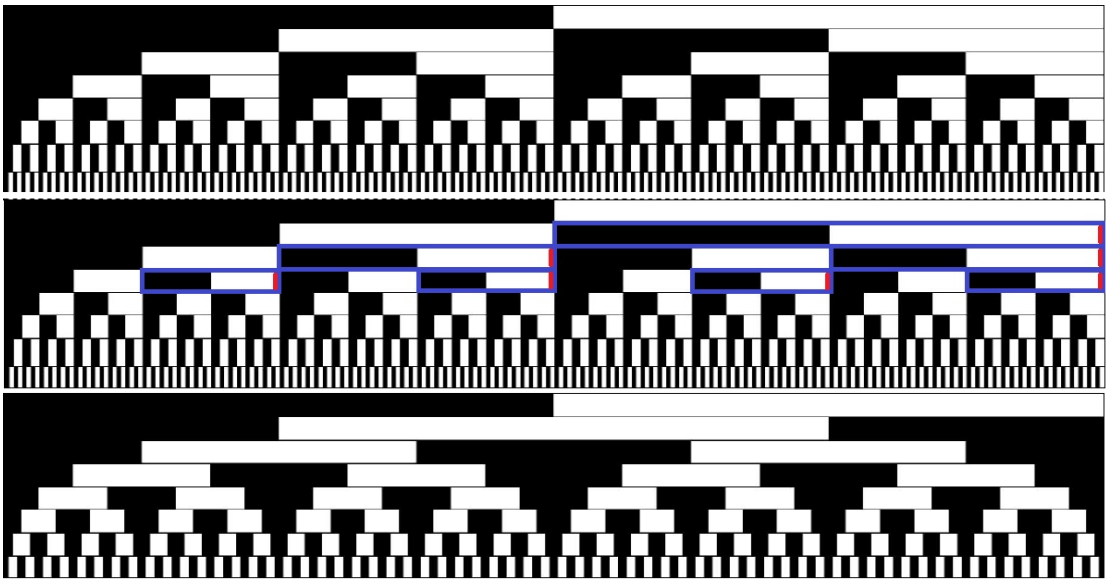


Figure 2.2. A simple binary code (top) compared to Gray code (bottom). Notice that when certain sections (for example the blue boxes in the middle panel) of a simple binary code is reflected in a specific manner (indicated in red on the right of each section), a binary Gray code is produced.

Spatial encoding patterns are generally more complex than those in a temporal encoding sequence. The patterns can either be constructed for the sole purpose of distinguishing regions of the scene, that is, a specific design without the use of any mathematical encoding theory, or one that is based on De Bruijn sequences [25, 27, 28, 29]. A commonality of many spatial encoding strategies, in particular those based on De Bruijn sequences, is that the codeword for each encoding region is not unique on its own (see Figure 2.3), but when combined with the codeword for neighbouring encoding regions, the resulting subsequence becomes a unique codeword. This dependency on neighbouring regions means a misdetection on a small part of the pattern during encoding can cause a greater amount of data loss in the decoding phase.

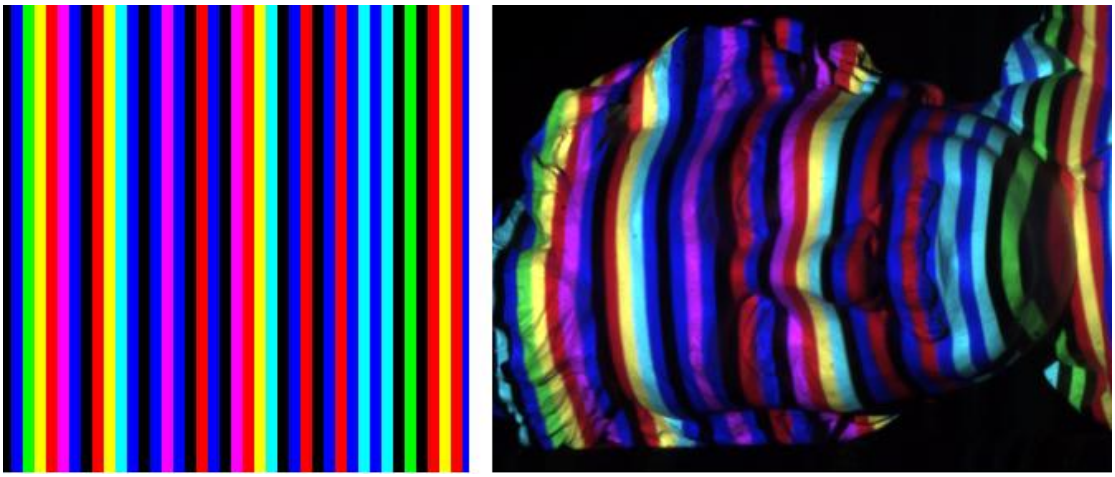


Figure 2.3. Section of a spatial encoding pattern (taken from [28]) based on a De Bruijn sequence with subsequence length of 3 (left), and the same pattern projected onto an object (right). Notice that the codeword for each encoding region (vertical stripes) is not unique on its own. For example, the colour yellow is used in four encoding regions. If however, the encoding regions are examined in groups of three, none of the combinations are repeated.

Besides the two main categories, there are also direct codification strategies that use a large number of either colours or grey levels so that the entire scene can be encoded in one pattern [27] and all encoding regions can be identified independent of all the other regions. Direct codification strategies can be very challenging to implement due to the high sensitivity to noise as well as the intrinsic colour of surfaces in the scene [27].

2.3. Camera Calibration

2.3.1. Coordinate Systems

In the mathematical model of projecting a 3D point in world coordinates onto the image plane of a pinhole camera, the position of the point on the image depends on: (1) where and in what orientation the camera is relative to the point, and (2) the internal geometry of the camera. Three coordinate systems involved in this model are: the world (or scene) coordinate system, the camera coordinate system, and the image coordinate system (see Figure 2.4).

The world coordinate system can be chosen in any convenient way. The coordinate of an object in the room is a relative quantity, it depends on where the origin is and how the axes are orientated, however, the relative positions between objects remain unaffected. The choice of the world coordinate system therefore makes no difference to the final image taken by the camera.

The camera coordinate system is a three-dimensional coordinate system centered on the focal point of the camera. In this project, the *Z*-axis is always the optical axis of the camera, and the *X* and *Y* axes are positioned accordingly. The transformation from world coordinates into camera coordinates is linear, and can be calculated explicitly with the aid of a registration cube, or using calibration patterns.

The image coordinate system is a 2D system that describes the final location of a point on the image. Image coordinates are the projection of 3D points in camera coordinates onto the image plane using the pinhole model (explained further in the next section).

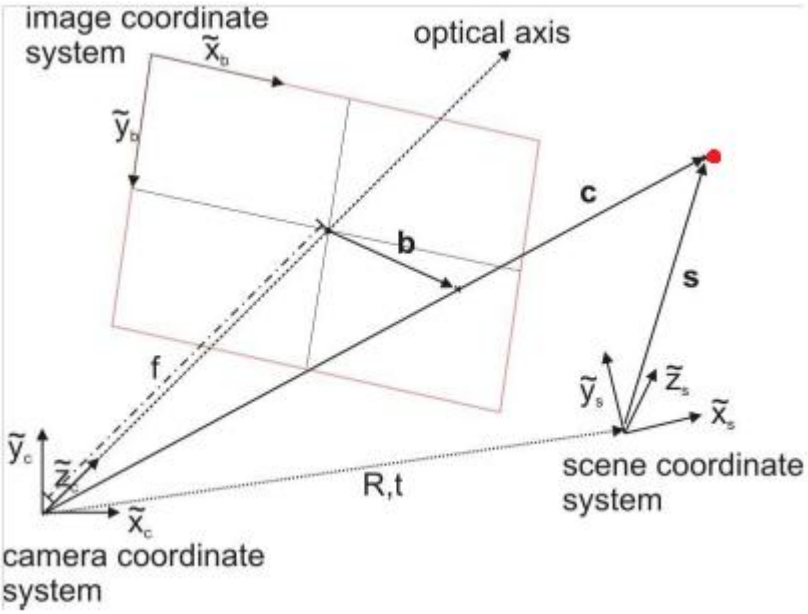


Figure 2.4. Coordinate systems. The bald red point can be shown in three different coordinate systems, in world coordinates (vector **s**), in camera coordinates (**c**), or in image coordinates (**b**)

2.3.2. Camera Model

The objective of camera calibration is to determine the parameters that describe the mapping between 3D world coordinates and the 2D image coordinates [30, 31]. The set of parameters that maps 3D camera coordinates onto 2D image coordinates is known as the intrinsic parameters. Intrinsic parameters represent the camera's internal geometric and optical characteristics, these include: focal length, principal point, skew coefficient, and distortions [25, 33]. Parameters that determine the mapping from world coordinates onto camera coordinates are the 6 extrinsic parameters, 3 for the Euler angles for rotation around the *x*, *y*, and *z* axes of the world coordinate system, and 3 for translations in the *x*, *y*, and *z* directions, respectively [32, 33].

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \text{where} \quad \mathbf{A} = \begin{bmatrix} f_x & \alpha_c & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

For a given scene point in 3D space, *u*, *v* are image coordinates in pixel units showing where on the 2D image this particular scene point would appear. *X*, *Y*, and *Z* are the 3D coordinates of the scene point in world coordinates. In the extrinsic matrix, vectors $\mathbf{r_1}$, $\mathbf{r_2}$, and $\mathbf{r_3}$ form columns of the rotation matrix that describes the optical

axis orientation (look angle) of the camera with respect to the world coordinates, and the fourth column **t** is the 3D displacement vector of the camera focal point and the origin of the world coordinates. In the intrinsic matrix A, f is the focal length of the camera, it is presented in pixel units and appears twice in the intrinsic matrix to deal with non-square pixels (i.e. aspect ratio), C_x and C_y are the position in pixel unit, of the central point of the image (called the principal point), where the origin is the top left corner of the image by convention. α_c is the skew coefficient which defines the angle between the x and y pixel axes, for most cameras this angle is close to 90 degrees leading to a zero value (or very close to zero), that is why in some calibration model the estimation for skew coefficient is skipped.

If X, Y, and Z are row vectors containing in world coordinates, the x, y, and z components of surface points of a 3D object, equation 2.4 then describes the perspective projection of a 3D object. Perspective projection is one of the most important underlying principles in Augmented Reality. It is the process of determining how a 3D object would appear on camera, given the intrinsic parameters of the camera, and the position and orientation of the camera relative to the object. It is important to note that while projecting a given 3D scene coordinate onto the image plane is straight forward using 2.4, the reverse (i.e. 3D vision, the recovery of 3D coordinates from image coordinates) cannot be achieved using this equation alone.

2.3.3. Calibration Theory

The camera was calibrated using Bouguet's Camera Calibration Toolbox [34]. The toolbox is largely based on Zhang's method [33]. Zhang's method begins with the estimation of the homography relating image points and points on the calibration plane, the closed-form solution for the intrinsic and extrinsic parameters are then calculated from the homographies. The radial distortion coefficients are then solved using linear least-squares, and finally, every parameter calculated is refined through maximum likelihood estimation.

The specific method uses 2D external patterns in the form of checkerboards to provide the set of points in world coordinates. The camera model described by 2.4 can be simplified if the $Z = 0$ of the world coordinate system is chosen to coincide with the checkerboard, in this case, we may drop the third component from the vector for the scene point, and the third column of the 3 by 4 extrinsic transformation matrix.

Furthermore, let

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} \quad (2.5)$$

so (2.1) becomes

$$s\mathbf{m} = \mathrm{H}\mathbf{M} \quad \text{where} \quad \mathbf{m} = [u, v, 1]^{\mathrm{T}}, \quad \text{and} \quad \mathbf{M} = [X, Y, 1] \quad (2.6)$$

Matrix **H** is known as the homography, which relates 3D scene points and the 2D image of the scene. The homography is estimated in the calibration toolbox using a technique based on a maximum likelihood criterion [34].

Because rotation matrices have linearly independent unit vectors as columns, vectors **r** have the properties:

$$\mathbf{r}_1^T \mathbf{r}_2 = 0, \quad \text{and} \quad \| \mathbf{r}_1 \| = \| \mathbf{r}_2 \|$$

From (2.5) and the above conditions yield two constraints on the intrinsic matrix A:

$$\mathbf{h}_1^T \mathbf{B} \mathbf{h}_2 = 0, \quad \text{and} \quad \mathbf{h}_1^T \mathbf{B} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{B} \mathbf{h}_2, \quad \text{where} \quad \mathbf{B} = \mathbf{A}^{-T} \mathbf{A}^{-1}$$

Each unique view of the checkerboard pattern (referred to as a calibration pattern) can yield two constraints on the intrinsic parameters. **B** is a 3x3 symmetric positive definite matrix, and since **B** has six degrees of freedom, at least three calibration patterns are required to calculate **B**. With three calibration patterns, three unique **H** may be obtained to form a system of six equations for the six unknowns of **B**, once **B** is estimated the intrinsic matrix can be found [33].

2.3.4. Calibration Procedure

A 9 by 7 checkerboard was used for calibration. The calibration board was made of printed checkerboard pattern pasted on a flat panel, and each square measured 30mm x 30mm. The calibration board was held in front of the webcam over a range of distances and rotation angles (see Figure 2.5). For each calibration board orientation, a 1200 x 768 pixels image was acquired and saved as a jpg file. After at least ten calibration patterns were acquired, the calibration toolbox was run under Matlab to process the saved calibration images.

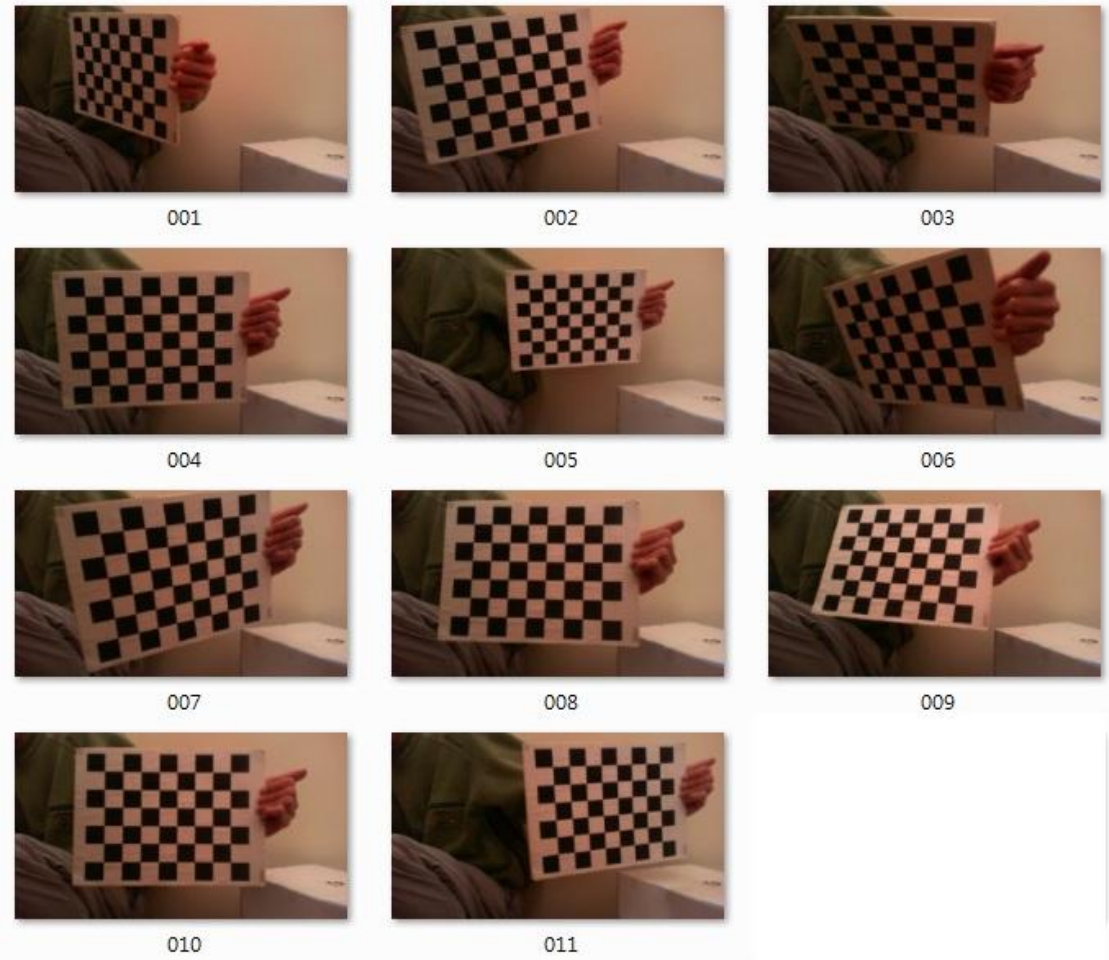


Figure 2.5. Calibration using a checkerboard pattern.

The calibration toolbox did not automatically detect the grid corners. For each image the four extreme grid corners must be selected manually, the toolbox could then search for all remaining grid corners automatically (see Figure 2.6) using a built in corner detecting algorithm. The grid corners found were in 2D image coordinates (*u*, *v* in equation 2.4). A vector (*X*, *Y*, and *Z* in equation 2.4) in the world coordinate system was assigned for each grid corner. The first clicked grid corner was used as the

origin, and the coordinates for all other grid corners were found by adding the appropriate integer multiples of the grid corner spacing in *X* and *Y* (with *Z* = 0 on the board, see section 2.3.3).

The selection of the extreme corners was done by clicking of the mouse, the corner detector helped minimize the clicking error, as long as the mouse click was no more than 5 pixels away from the true corner. The first clicked point was automatically chosen to be the origin of the world coordinate system. The choice of the first point is of particular importance in a multi-camera system. In multiple camera calibration, each calibration board orientation has a set of corresponding calibration images (one image for each camera), and the same first point must be clicked on all images across the set to accurately estimate the relative positions of the cameras. Figure 2.7 is a visualization of the extrinsic camera calibration result.

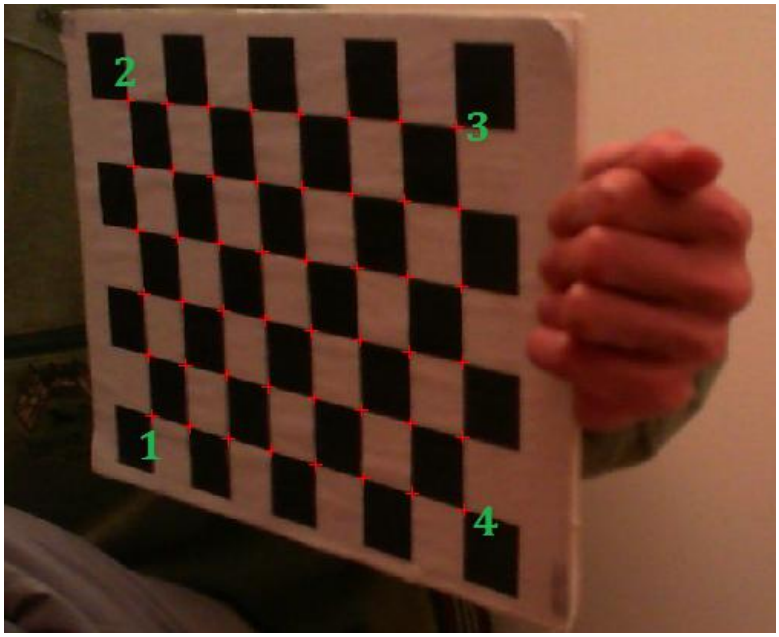


Figure 2.6. Auto detection of grid corners (red markers) after the four extreme grid corners (red markers with numbers indicated right beside) are given.

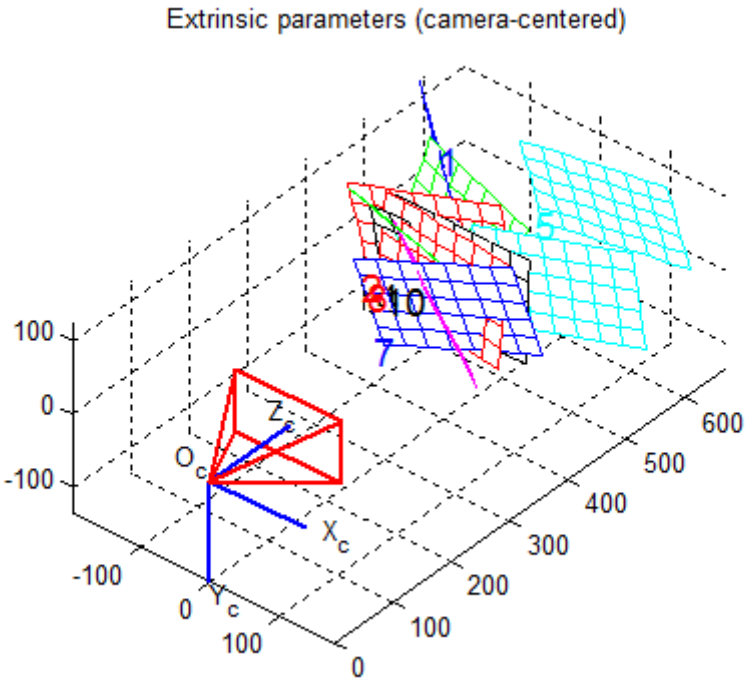


Figure 2.7. For each calibration image, a set of extrinsic parameters can be estimated. enabling the recovery of checkerboard location at the time of image acquisition. In the figure, the axes of the camera coordinate system are shown in the blue with the point O_c being the origin as well as the focal point. Colored grids numbered 1-11 are the estimated locations of the checkerboard featured in the Figure 2.5

2.4. Projector Calibration

LCD projectors work by splitting a strong white light source into the primary colours through a prism, then onto small LCD panels. These panels control the amount of red, green, and blue light that can pass through each pixel on the panels. When the light passing through the LCD panels are recombined, a 2D image is formed. This location inside a projector is analogous to the image plane of a camera, and for the sake of simplicity, it will be known as the image plane of the projector from here on. From the image plane, the recombined image is then projected onto the scene through divergent lenses.

A projector is essentially the reverse of a camera [25]. A camera converges light from the scene and forms an image that is the projection of the scene onto its 2D image plane, this is just like a projector (up to the recombination step), except the light forming the image propagates in the opposite direction and towards the scene.

Mathematically, equation 2.4 is therefore also valid for projectors.

The calibration procedure for a projector is in principle similar to that of a camera, the key differences lie with the design and function of the calibration board and the extra step required in the software. Projectors cannot capture images of a checkerboard, so instead a checkerboard pattern is projected onto a plain board. Grid corners of the projected checkerboard are used as scene points for calibration. Unlike the printed checkerboard used in camera calibration, the dimensions of the projected checkerboard pattern depend on the relative positions of the calibration board and projector. The 3D position of the grid corners, in the world coordinate system therefore cannot be simply assigned for like in camera calibration.

One way to calculate the grid corners is to attach points of reference to the calibration board so the position and orientation of the calibration board is known [25]. This is essentially a camera extrinsic calibration. From the knowledge that the grid corners lie on the plane, 3D coordinates of the projected grid corners can be found using either a ray-plane intersection or a perspective transformation based method. At this point, there are two choices of coordinate systems for the 3D coordinates:

1. In world coordinates - choose one of the four reference points (the four fiducial markers in Figure 2.8) as the origin, let the calibration board coincide with the $Z = 0$ plane, assign coordinates to the three other known reference points, then use the perspective transformation to calculate the corner coordinates of the projected checkerboard.

2. In camera coordinates - use a calibrated camera to look at the calibration board, calculate the extrinsic parameters from the four reference points using the calibration toolbox, this gives the translation and rotation of the calibration plane relative to the camera. Then either apply the transformation after calculating the world coordinates using option 1, or build direction vectors for 3D rays going through the grid corners, then find the ray-plane intersection for each grid corner.

While presenting the grid corners in camera coordinates requires an extra step in the calculation, it has the added advantage of providing projector position and orientation relative to the camera. This is particularly useful in structured light vision methods (see the section 2.1). Table 2.1 is a summary and comparison of the camera and projector calibration procedures.

	Camera Calibration	Projector Calibration
Mathematical Model	Equation (2.4)	Equation (2.4)
Pixel sizes of grids (u, v) on the image plane	Grids have different sizes	All grids have the same size
Dimension of grids (X, Y, Z) in the actual scene	Grids on the printed checkerboard all have the same size	Grids on the projected checkerboard have different sizes
Determining grid corners in image coordinates on the image plane	Measured directly from images using corner detector	Assigned based on the pixel dimensions of the projected checkerboard
Determining 3D coordinates of scene grid corners	Let $Z = 0$, first clicked point being the origin, coordinates assigned to the other points because checkerboard dimensions are known	Calculated from the image coordinates of: 1. fiducials, and 2. grid corners measured by a calibrated camera
Coordinate system in which scene grid corners are presented	World coordinates, $Z = 0$ along the surface of calibration board, origin chosen by first click rule	Camera coordinates, camera focal point is the origin, Z axis along the camera optical axis

Table 2.1. Comparison of camera and projector calibration.

Projector Calibration Procedure

A plain 40 cm by 40 cm board with fiducial markers on the four corners was constructed. The fiducials used were in the form of 2 by 2 checkerboard patterns, they were pasted onto the corners of the board to form a 30 cm by 30 cm square between the grid corners. The remaining area of the board was covered in white (see figure 2.8).

The projector was controlled by a Matlab script using functions in the Psychophysics Toolbox. The Psychophysics Toolbox is a third party Matlab toolbox for generating and presenting visual stimulus in cognitive science studies [35]. Its core routines provide user access and control to all aspects of the graphics and display hardware [35, 36]. The toolbox also comes with an extensive range of functions for stimuli generation, both visual and audio [36]. While the toolbox was originally designed for use in experimental cognitive science, it is capable of interfacing between Matlab and computer displays in most vision research.

A 1024 by 768 image matrix representing a 8 by 8 mono-colored checkerboard pattern in the middle and white background was generated in Matlab. The image was projected onto the calibration board using the *Screen* function in the Psychophysics Toolbox. The function *Screen* converts a Matlab image matrix into input signals for computer displays, and then passes the signal onto the appropriate device according to the user specified *screen number* (the Psychophysics toolbox automatically detects all display units connected to the computer, and numbers them in integers). Images of the projected pattern were captured for a range of board locations and poses. The method for obtaining the image coordinates of the projected grid corners was identical to the corner extraction procedure in camera calibration: manually select the four extreme corners followed by automated corner detection of the rest. The four reference points for board location and orientation tracking were also manually found with the aid of the corner detector, the image coordinates were then put through the extrinsic calibration function of the calibration toolbox, to determine their 3D locations in camera coordinates.

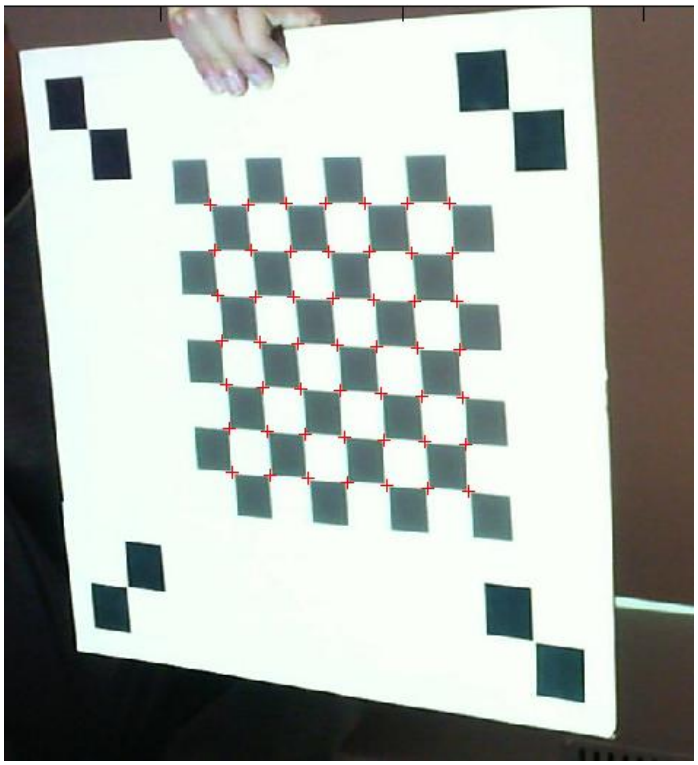


Figure 2.8. The projector calibration board featuring four fiducials on the four corners, checkerboard projection in the middle, and the grid corners (red crosses) found by the corner detector.

2.5. Temporal Encoding Implementation

The Matlab code used for generating the temporal encoding sequence was primarily based on Lanman & Taubin (2009) [25] with changes made only to the capturing method of the encoded scene.

Lanman and Taubin's Gray code implementation [25] consisted of all of the patterns that would feature in a binary sequence, as well as the inclusion of the binary inverses of each binary pattern (see Figure 2.9). The length of the encoding sequence was determined by the resolution of the encoding projections. For example, the length of binary codeword would have to be at least 10 (i.e. 10-bit), to encode an array of 2^{10} = 1024 unique segments, which also means an encoding sequence of at least 10 images would be needed. If the binary inverses were added, the encoding sequence length would double. All-on and all-off images (refer to the first two patterns in Figure 2.9) were introduced in Lanman and Taubin's implementation to improve decoding accuracy. In decoding, the software assigned every pixel of each captured frame as either lit or not-lit by examining the r-g-b values at that pixel. A threshold value in intensity was therefore required to complete the determination process. A universal threshold value for this had a major drawback: due to geometry of the surface been scanned, the intensity of illumination varies across the scanned surface [25]. This meant an un-lit pixel could be misassigned as an "on" pixel due to scattering of light, and conversely, a lit pixel could be interpreted as an "off" pixel because the surface curved away from the incoming light direction (see Figure 2.10). The all-on/all-off patterns allowed the local illumination intensity to be taken into account when deciding if a pixel was on or off.

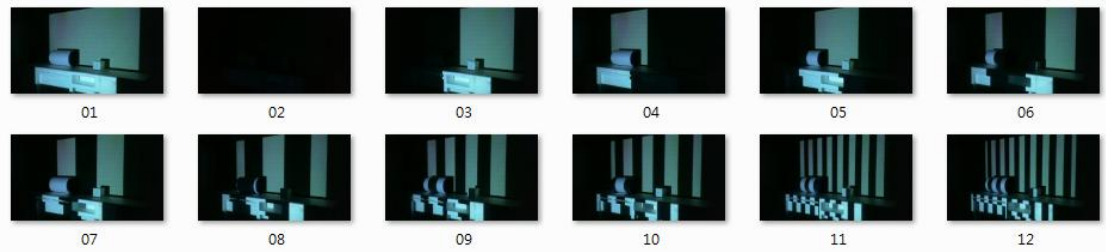


Figure 2.9. First 12 frames of a Gray code encoding sequence, notice that the number of stripes increases and their width halves on every odd numbered frames, these are the Gray code sequence. On every even numbered frame, the pattern is the reverse of the previous (odd numbered) pattern, these are the binary inverses of the Gray code sequence.

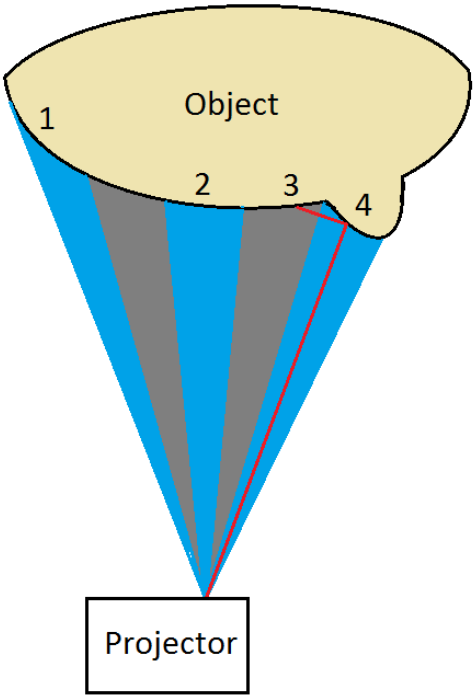


Figure 2.10. Example of the encoding process, blue represents the path traversed by the lit sections of the encoding pattern and grey represents the un-lit sections. Region 1 on the object surface will have lower illumination intensity than region 2 due to the curvature. Region 3 will appear brighter than the other un-lit regions due to scattered light from region 4.

2.6. Spatial Encoding Implementation

Despite the large amount of available literature on spatial encoding structured light, at the time of this project, a ready-to-use spatial encoding structured light program for Matlab was not available. While some utility functions from Lanman and Taubin's structured light toolbox, such as those used in triangulation, could be reused for the spatial encoding technique, the majority of the method still had to be programmed. Implementing an efficient and elegant program, particularly for the decoding part, would require a strong background in combinatory theory. This was considered too difficult for the time frame of this project, and a "keep it simple" approach was followed instead, disregarding the importance of efficiency.

2.6.1. Codeword creation

A De-Bruijn sequence generator for Matlab [37] was used for the generation of the codeword for the encoding sequence, the function used two user specified inputs:

the number of characters, and the sub-sequence length, to generate an interleaved pseudo-random De-Bruijn sequence and in the format of an integer array [37]. The numbers were than translated into colours, for example, 1 for the colour red, 2 for green, and 3 for blue. One drawback with this generator, for the purpose of spatial encoding at least, was the immediate repetition of any number in the sequence, For example, one of the six possible De-Bruijn sequences for 3 characters and sub-sequence length 2 was "121332231". In a codeword, a segment of the same number repeating itself (“33” and “22” in the previous example) would translate the whole neighbourhood of encoding regions into the same colour. This became an issue because it was difficult for the software to distinguish whether a stripe on the camera image was really one stripe or multiple neighbouring stripes all with the same colour. In the last example, the “3322” segment should translate to “blue-blue-green-green”, but it could easily have been mistaken as (but not limited to) “blue-green”, “blue-blue-green”, or “blue-green-green”.

The computation in the original code used a brute force approach. For a De-Bruijn sequence using n number of characters and k subsequence length, a list of every possible permutations of length k, using only the integers within the range 1 to n, was generated. From the list, one subsequence is randomly chosen at a time, and its first k-1 elements were compared to the last k-1 elements of the current overall sequence. Call the described process Step 1 for convenience and let m be the current overall sequence length. Step 1 was repeated until a match was found and in which case, the matching subsequence was deleted from the list and added to the overall sequence (m to m+1). The software would then loop back to Step 1, to search for the next matching subsequence for the updated overall sequence (now m+1 in length). In the case when no matches could be found for m+2, the program would backtrack by removing the last matched subsequence (m+1 to m) and search for an alternate. Call this Step 2. Step 2 would be repeated if no alternative could be found for m+1, and it was not unusual for the program to keep backtracking until m = 0, in which case a new starting subsequence would be randomly chosen from the list.

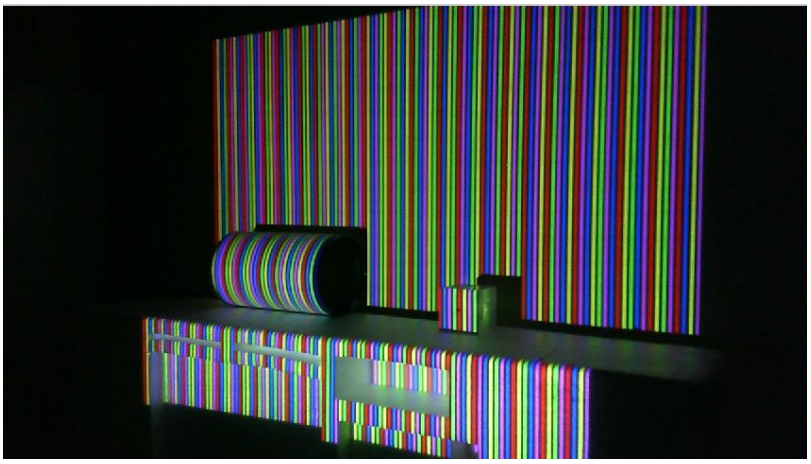
The De-Bruijn generator was modified to generate a De-Bruijn like sequence that differed from the original in not having any consecutive appearance of any one number. None of the core steps, such as Steps 1 and 2 described in the previous paragraph, needed to be modified. The generation of the list of subsequences was modified. Instead of finding every possible combination, the criterion was changed to allow only combinations in which any one number appeared at most once. The Matlab functions *nchoosek* and *perm* were used for this purpose. *nchoosek* by

default finds all combinations of length k, from n things, without using any one element more than once per combination. *perm* tabulates all possible permutations of a list of numbers, this function was used on each combination found by *nchoosek* to generate the full, new list of subsequences under the modified criterion.

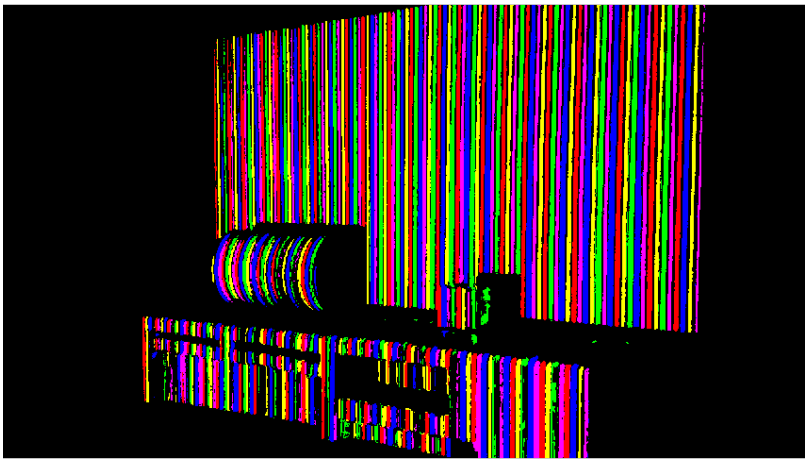
The resolution of the webcam used in this project was not high enough to make full use of the projector resolution of 1024-by-768 pixels. The encoded scene occupied from one-third to half of the camera image, which had a resolution of 1280-by-720 pixels, this meant encoding patterns would lose around half of their resolution during capturing of the scene image. The smallest feature on the encoding pattern therefore had to have at least 2 pixels in both dimensions to have a reasonable chance of being seen in its entirety. To avoid misdetection, the stripes on the encoding pattern were set to 5 pixels wide.

2.6.2. Codeword Extraction

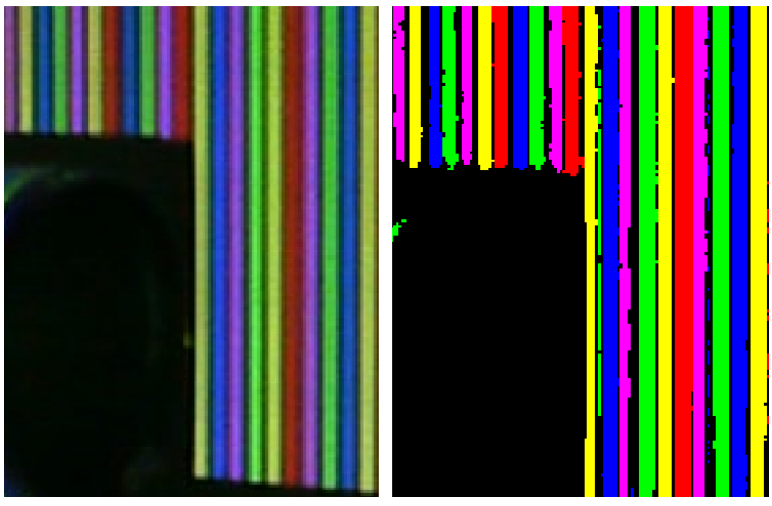
The image of the encoded scene was saved in RGB format, each colour channel of a given pixel was stored as an 8-bit unsigned integer (UINT8 format, range between 0 and 255). The colour data for every pixel was converted from RGB, based on a series of colour selection criteria, into integers 1 to n for pixels that passed the selection, where n is the number of colour used, and 0 for those that did not meet any of the criteria. The mean value of the three colour channels was calculated for every pixel, a lower threshold was set based on the mean, and any pixel with a value below the threshold was deemed part of the un-encoded regions or background and was assigned the value 0. For all pixels that passed the brightness filter, ratios of colour channels (e.g. r/g, g/b) were calculated. The colour channel ratios were used for determining the colour of the pixel. As an example, if r/g = 2.23, r/b = 1.92, and b/g = 1.16, the colour assignment for that pixel should be red or the value 1, because the red channel was significantly more prominent than both the green and blue channels. The output from the described colour assignment was a 1280-by-720 matrix that had the same number of representations of colours as those in the original encoding pattern. A global set of criteria was used for colour determination. Colour identification errors most commonly occurred around the edges of a stripe, examples of this can be seen in Figure 2.11c, the fourth and tenth strips from the left were magenta and yellow respectively (Figure 2.11c, right), however, the left edges of these stripes were identified as blue and green respectively (Figure 2.11c, left).



(a) Encoded scene image



(b) Colour identified from scene image



(c) Colour identification error, leftt: image of the scene, right: identified colours.

Figure 2.11. Identification of 5 colours (Red, Green, Blue, Yellow, and Magenta) used for encoding, they appeared in subsequences of 4 with a black stripe interleaving every neighbouring pair of coloured stripes.

2.6.3. Decoding

Due to the short length of the codeword (less than 200 elements), the decoding was done through a "brute force" method, by searching for each observed subsequence in the original encoding sequence. The codeword for each row of the image pixels were compared to the original codeword used to generate the encoding pattern. Segments of the codeword indentified from the image were systematically extracted, 4 elements at a time. A vector **v**, identical in length to the camera captured codeword, and containing the location of each codeword element within the original codeword was calculated. Let m be the length of the imaged codeword, n the length of the original codeword, y the index for elements of the imaged codeword, and x the index for elements of the original codeword, the steps for finding correspondence between the two code-words were as follows:

1. The first 4 elements of the codeword segment ($y = \{y_0, y_0+1,..., m-1, m\}$) were extracted to form a subsequence, a matching subsequence was searched for in the original codeword. Let $x = x_0$ be the vector index of the first element of the segment in the original codeword that matched the subsequence, the location of this subsequence would be x_0, x_0+1, x_0+2, x_0+3.

2. Compare every subsequent element of the segment (that is, $y = y_0+j$, where y0 is the vector index of the first element of the segment from the captured codeword, and $j = \{4, 5, ..., m- y_0\}$) to the corresponding elements of the original codeword (i.e. $x = x_0+j$), proceed until either (a) a mismatch was encountered or (b) all elements of the segment were matched to the original codeword. In the latter case, skip Steps 3 and 4.

3. If a mismatch between the imaged and original codeword occurred, it was likely due to a jump in the imaged codeword caused by steep surfaces. In this situation, the segment starting on the element mismatched would be extracted by setting the current y to y_0 and return to Step 1.

4. If a new starting point in the original codeword could not be found, the mismatched element seen in Step 3 was probably the result of either an error in colour identification, or the encoding region was too narrow to be encoded by a complete subsequence of 4 coloured stripes. This element in the imaged codeword would be ignored (value 0 assigned to it) and the procedure looped back to Step 1.

5. Every element of the imaged codeword should now have either a corresponding element in the original codeword, or been flagged as unidentifiable.

2.7. Structured Light Reconstruction Results

A test scene was constructed out of simple 3D shapes, these included a cylinder, a square box, and a polystyrene container with rectangular blocks of varying sizes cut off (see Figure 2.12). All of the objects in the scene were made white to minimize the effect of intrinsic surface colouration on spatial encoding. Both the temporal and spatial encoding techniques were used to reconstruct the scene.

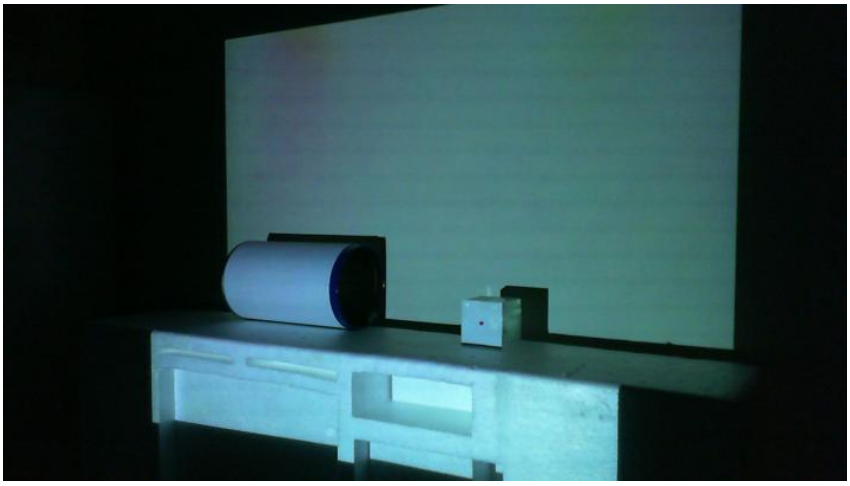
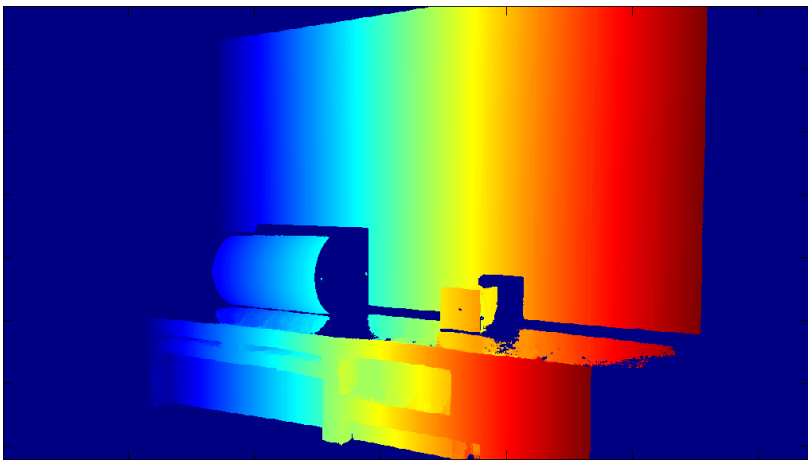
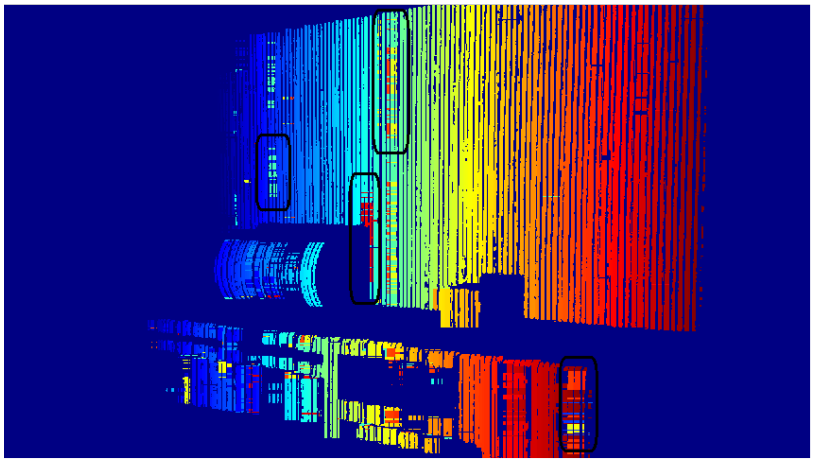


Figure 2.12. First test scene for the structured light 3D vision methods.

The recovered projector column numbers for both encoding techniques are shown in Figure 2.13. The three rectangular cut-out regions, or gaps for short, along the top edge of the polystyrene box were orientated parallel to the projector's optical axis, it was therefore difficult for these parallel surfaces to be encoded by the light field. The spatial method failed to form a valid codeword-segment for these regions due to insufficient illumination. This was evident in the encoded scene (see Figure 2.11a). The temporal method was able to both encode and decode the majority of the cut-out region (see Figure 2.13a), however, upon examining the reconstructed point cloud, not all of the decoding was correct. For instance, the bottom surface of the third cut-out (thickest of the three) in the reconstructed point became transverse to the projector optical axis when in reality it was a parallel surface. This could have been the result of encoding pattern being reflected off the back wall of the cut-out. Decoding errors were common in the spatial method (for example the enclosed regions in Figure 2.13b), most of which were due to colour identification errors previously described.



(a) projector columns recovered from the temporal method.



(b) projector columns recovered from the spatial method, black boxes showing regions where decoding error occurred.

Figure 2.13. Decoded projector column indices displayed in colour, ranging from dark blue for column 1, to dark red for column 1024

The reconstructions of the scene are shown in Figure 2.14, features on the polystyrene box can be seen in detail in the temporal method. In the reconstruction using the spatial encoding technique, only large flat surfaces could be successfully reconstructed.

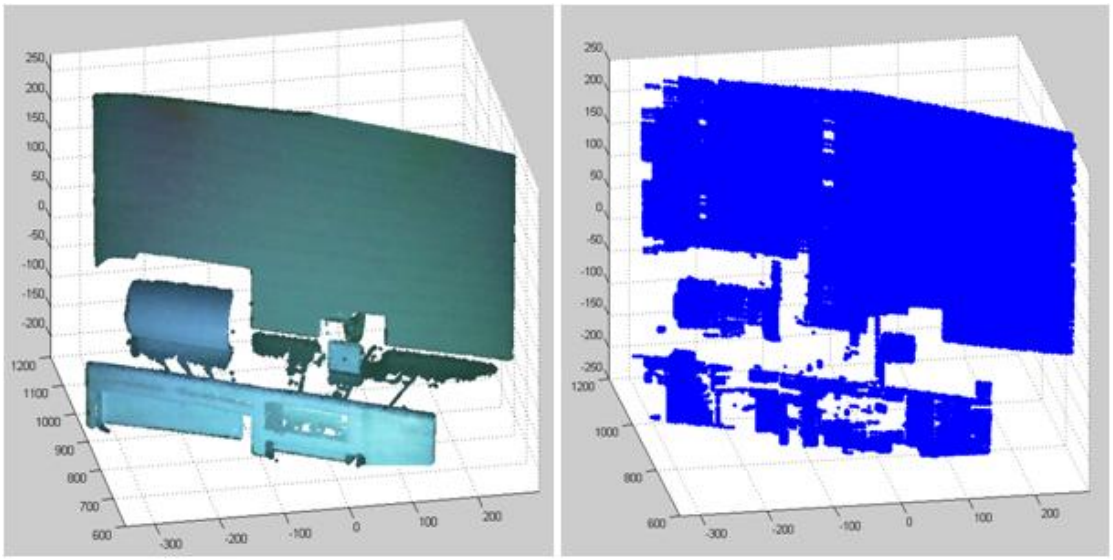


Figure 2.14. 3D scene reconstruction using temporal (left) and spatial (right) encoded structured light

2.8. Summary

A 3D structured light system was setup and tested. Both the temporal and spatial encoding strategies were trialed. Temporal encoding technique based on Lanman and Taubin (2009) [25] required a sequence of 42 images to encode the scene, the encoding time requirement was compensated by the robustness of the method under non-ideal lighting conditions. The spatial encoding technique implemented required only one image of the encoded scene to produce a 3D reconstruction, however, the simple colour recognition procedure implemented was highly susceptible to detection error. The colour stripes were made wider and interleaved in order to reduce detection error, however, this also lowered the spatial resolution.

The two structured light methods differed only in their encoding and decoding techniques. If the techniques were on-par in decoding accuracy and resolution, then they should, in principle, have the same 3D reconstruction error. In reality however, the simplified spatial encoding implementation could not achieve the same level of accuracy and resolution as the temporal method. The temporal encoding technique was therefore deemed the better choice for testing the accuracy of the structured light method. Figure 2.15 is a qualitative preview of the capability of the 3D vision method using structured light temporal encoding.

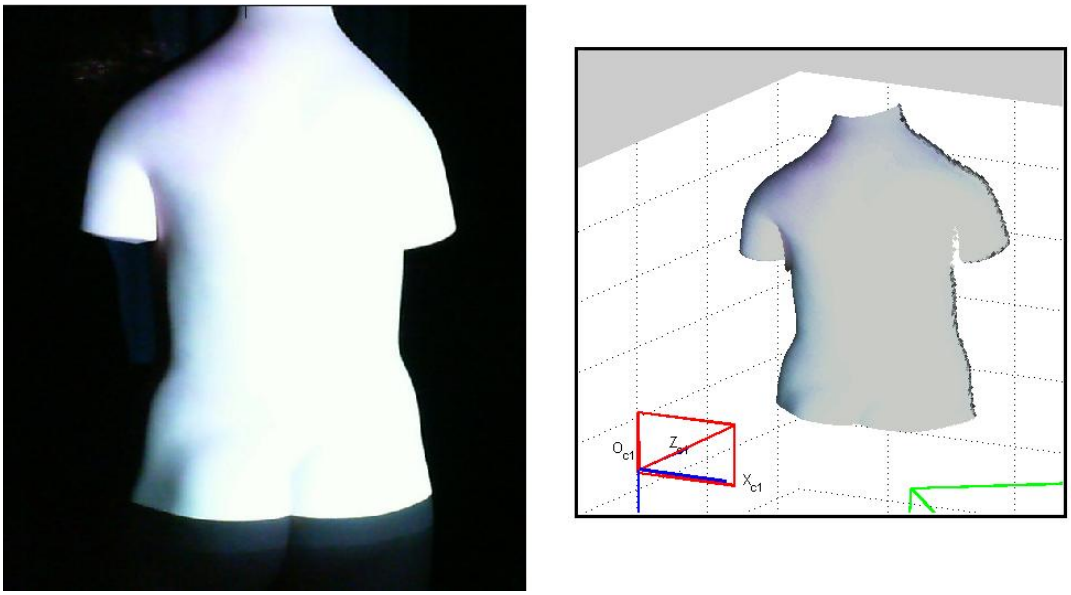


Figure 2.15. The back of an anatomical phantom (left) and its 3D reconstruction using temporal structured light (right).

Chapter 3

Structured Light Accuracy

Offsets between captured 3D patient surfaces and CT data sets can have any or all of the following four origins: (1) Patient movement, change in posture, (2) misalignment/error in patient setup, (3) Inaccuracy in camera tracking, (4) Inaccuracy in the measured 3D surface. Out of these four types of offsets, (1) and (2) is what the 3D vision system sets out to detect i.e. the true offset of patient position and pose, (3) and (4) are errors that should be minimized.

Inaccuracy in camera tracking will generate error in the transformation between camera and world coordinates, this error will in turn cause the planned patient surface contours to be misplaced in camera coordinates.

Inaccuracy in the measured 3D surface refers to any offsets caused by the 3D vision system, measured when both type (1) and (2) offsets have been eliminated. Quantifying errors (3) and (4) separately was technically challenging. In the existing AR system, the tracking error (type 3) was thought to be only in the rotational parameters. It was estimated that the camera tracking procedure had rotational errors of 0.5^{o} in roll, pitch and yaw. To put this error into context, if a registration cube of 9.5 cm sides was positioned on the linac couch using the room laser setup approach [19], with an intentional 0.5^{o} offset in yaw, the maximum misalignment between the sagittal room laser and the z-axis of the cube would be 0.4 mm. Due to the finite laser beam width and temperature influences, it is difficult to directly measure errors of this magnitude. This is reflected on the 1 mm laser accuracy tolerance recommended by IPEM [38], and AAPM [39] for stereotactic procedures [39] that demand the highest level of accuracy.

In this chapter, error type (3) was estimated by examining the re-projection offset. Error type (4) was examined both through direct measurement and re-projection offset. In section 3.2, an experiment was designed to quantify the combined effect of errors (3) and (4), by using the 3D vision method to capture and reconstruct the camera registration device, and at the same time registering the camera.

It is also worth noting that, the terms extrinsic camera calibration, camera tracking, and camera registration are used synonymously, they all refer to the procedure of calculating the camera's extrinsic parameters from the corners of a checkerboard pattern with known dimensions (see section 2.3). The terms tracking board, calibration board, and tracking device all refer to the same board seen in Figure 2.8, which is analogous to the registration cube used in the original AR system. The board was previously used for projector calibration. In this chapter however, it was used as a test object for 3D reconstruction while simultaneously serving as the object for camera registration.

3.1 Capturing 3D surface of a plane

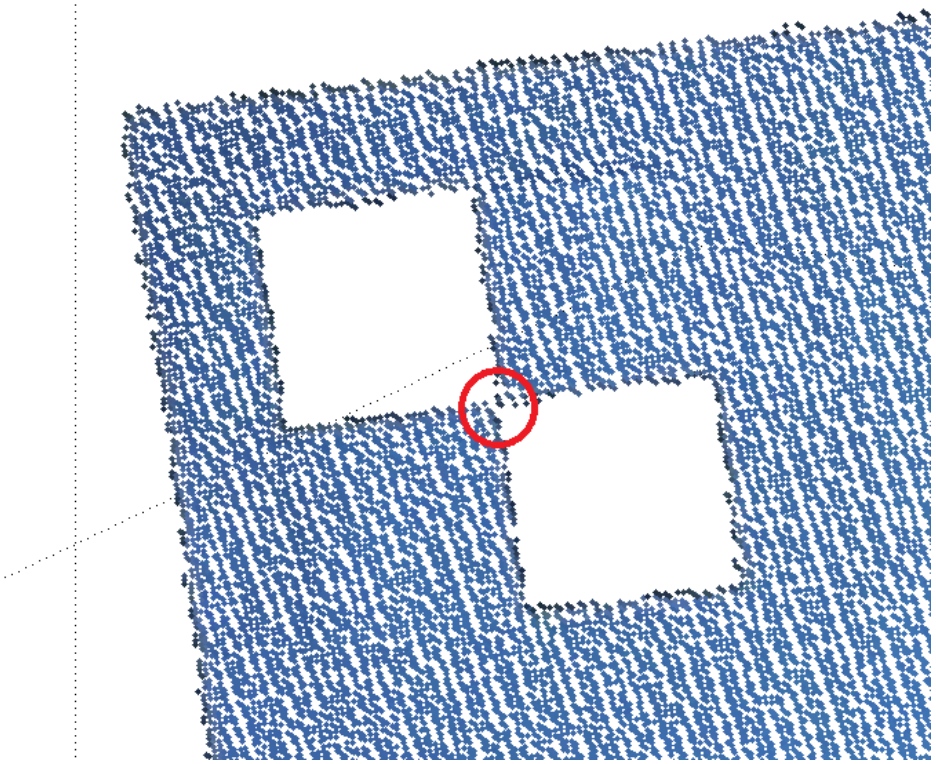
In this experiment, the 3D surface of a plane was captured. The goal was to characterize the accuracy of the structured light system by verifying the recovered surface with physically measured dimensions of the test object. A flat surface was the ideal test object for detecting and visualizing distortions in the 3D reconstruction. The accuracy determined from planar surfaces also applies to curved surfaces in the sense that every curved surface, however complicated, can be constructed by combining and truncating a set of planar surfaces with the appropriate orientations. The projector calibration board (see Figure 3.1b) was used as the planar object for the structured light system. The calibration board was chosen because it is in itself a camera tracking target (i.e. it was simultaneously used as both a tracking device and the imaging object). The captured image of the board was sufficient for the extrinsic calibration of the camera which is needed for virtual contour placement in AR. Instead of taking a CT scan to acquire surface contours of the board, coordinates representing the board surface were generated in the software using the measured dimensions with a perfect board surface flatness assumed. The optimum position and orientation of the plane was set to be its current position and orientation, which meant it was already perfectly aligned (since the board was itself the tracking target) and thus eliminating type (1) and (2) offsets.

3.1.1. Method

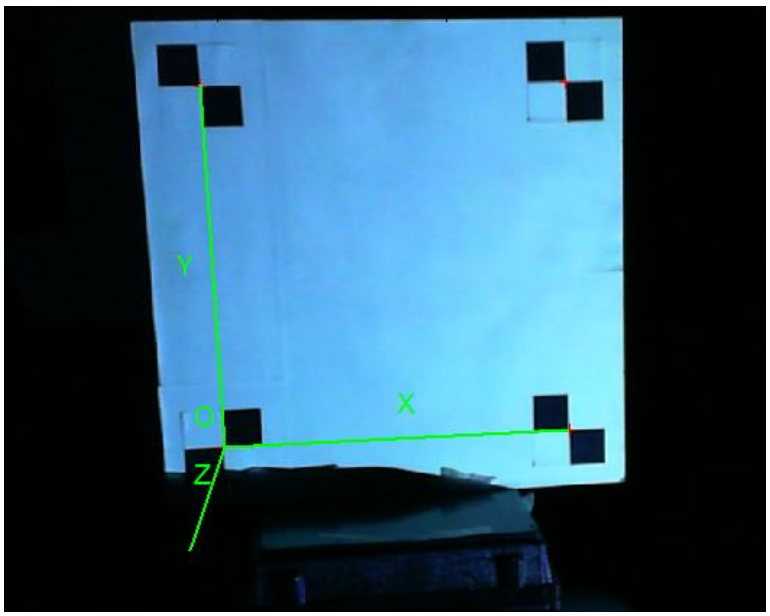
The camera and projector system was calibrated and fixed in one location throughout this experiment. The object plane was placed 1 m in front of the vision system, the room light was then switched off and the surface of the plane scanned using the temporal encoding structured light sequence. The first image of the encoding sequence (the all 'on' image) was also used for extrinsic camera calibration. The experiment was repeated for three different plane orientations: rotations of -10^{o}, 10^{o} and 35^{o} pan angle (about the y axis). A 13 degree tilt angle had to be added so the board could lean against the supporting stand behind.

The four fiducial corners on the each reconstructed 3D point cloud were identified manually. This task turned out to be simple because the square patches on each fiducial marker were black and could not be encoded and were shown on the reconstruction as square holes (see Figure 3.1a). Where the corners of two holes met (i.e. the region enclosed by the red circle in Figure 3.1a) was taken as one of the board corners.

The extrinsic calibration, or camera tracking, also allowed the coordinates of the fiducial markers to be determined. The two most significant differences between this and the structured light scan are: (1) the measurements done using the extrinsic calibration technique consisted of errors from the intrinsic calibration of the camera (i.e. determination of the focal point, principal point, distortion, etc.) and identifying the image coordinates of the fiducial corners, whereas in a structured light scan, on top of the two mentioned error sources, there is the addition of projector calibration errors. (2) Technical limitations aside, a structured light scan can be used on any surface, whereas the extrinsic calibration technique will only work if all four fiducial markers are present on the image, and coordinates can only be calculated for points that lie on the same plane as the fiducial markers.



(a)



(b)

Figure 3.1. (a) 3D plot showing the reconstructed point cloud zoomed in on the top left. The holes made the corner to be easily identifiable. (b) A photo of the calibration board, showing the fiducial markers, and the world coordinate system defined based on the fiducial corners.

3.1.2. Results

The coordinates of the corners found from the structured light scan and extrinsic calibration were compared and significant discrepancies were observed. The distances from the front of the camera to the fiducial markers on the board were measured directly with a tape measure and compared to the vision based measurements. There were discrepancies in measured camera distance between all three measuring techniques. To characterize the type of discrepancy between the techniques, an offset value was calculated at each corner, using measurements from all three plane orientations. Table 3.1 is a summary of the results.

	Corner 1	Corner 2	Corner 3	Corner 4
Directly measured and camera tracked	14 ± 1	13 ± 1	12 ± 2	12 ± 1
Directly measured and 3D reconstructed	-12 ± 8	-4 ± 5	10 ± 5	8 ± 3
Camera tracked and 3D reconstructed	26.3 ± 7.4	16.5 ± 4.0	2.6 ± 4.7	3.8 ± 2.8

Table 3.1. Difference in camera to corner distance measurements (in mm) between different techniques, values shown are the mean and standard deviation of measurements from the three plane orientations.

While the distances measured using the camera tracking/extrinsic calibration technique differed from those measured with a tape measure by 12 to 14 mm, the fact that all four board corners on all three plane orientations shared a similar offset from the direct measurements is an indication the measuring method suffered minimal distortion. The observed discrepancies could be attributed to how the origin was defined. The origin of the camera coordinate system that was used in both the extrinsic calibration method and the SL method was the camera focal point. In a direct measurement, the origin was a chosen point on the camera casing where the zero mark on the ruler sat. Since the focal point is usually not visible from the exterior of the camera, the origin definition was likely to differ between camera based measurements and the physical measurements, therefore discrepancies were expected.

In the structured light 3D reconstruction, the distance measurements of the corners were less consistent with the direct measurements. The offset ranged from 12 mm closer to the camera at corner 4 to an extra 10 mm away at corner 2. When compared to camera tracking/extrinsic calibration, the offset ranged from 26 mm at corner 4 to 2.6 mm at corner 2. In the construction of the calibration board, the fiducial markers were carefully positioned so their centers would form a 299 mm by 299 mm square, these can also be measured from the structured light reconstruction by finding the magnitude of the difference vector between each non-diagonal pair of corner coordinates (see Table 3.2).

	Side 1	Side 2	Side 3	Side 4
Orientation 1 (red)	302	304	298	306
Orientation 2 (magenta)	301	309	298	314
Orientation 3 (blue)	302	299	299	297

Table 3.2. Measurements of side lengths (in mm) of a 299 mm by 299 mm square using the structured light method, a visualization of the three plane orientations can be found in Figure 3.2, the sides are numbered in clockwise direction starting on the lower left corner (refer to the green markers in Figure 3.2). Colour references are made to the marker colours in Figure 3.2.

The structured light reconstruction had distortions in dimensions, it was most accurate on Side 3 for all three orientations, with less than 1 mm error, and most inaccurate on Side 4 with up to 16 mm in error. The distance measurements of the two corners (3 and 4) forming side 3 of the square were also in better agreement with distances found by camera tracking. This observation was in agreement with theory. Regardless of the structured light reconstruction accuracy, the perspective projection of the recovered corners back onto the image plane should always coincide with the pixel coordinates of the fiducial markers. That is, if the reconstruction made two points further away from the camera than they actually were (for example, corner 1 and 2, refer to Table 3.1), the distance between the pair of points would also be bigger (see Figure 3.3).

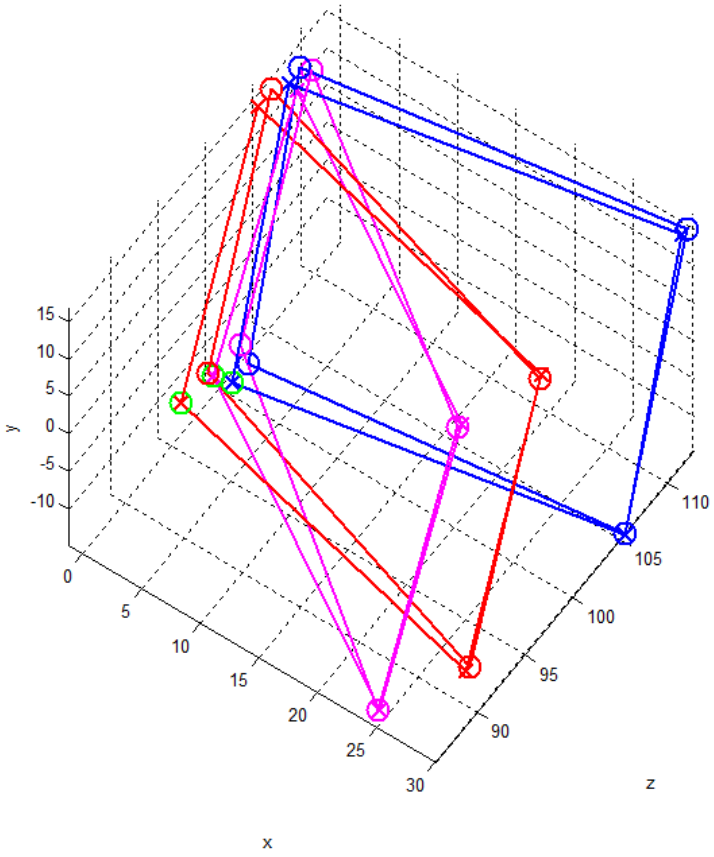


Figure 3.2. The four corners of the board in camera coordinates obtained from structured light 3D reconstruction (circles) and camera tracking (crosses). Each colour represents a different board orientation, all three results were obtained from the same camera/projector system. Scales are in cm.

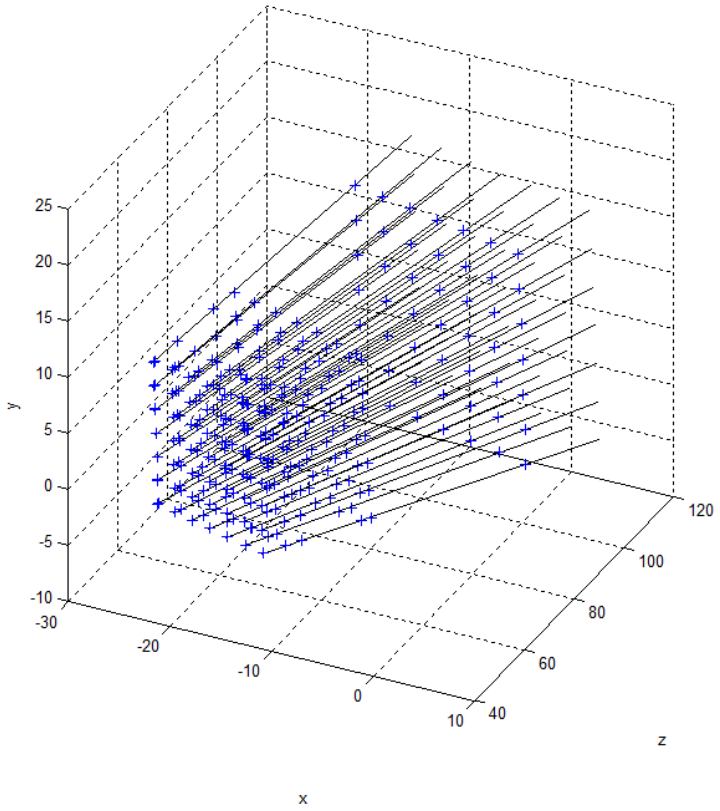


Figure 3.3. Relationship between camera rays (black lines) and points in the scene (blue crosses). In this example every camera rays passes through six grid patterns in the scene, while all six grids have different sizes, they should all project onto the same pixel coordinates on the image plane.

The structured light reconstructed corner coordinates were projected back onto the scene image using the intrinsic camera matrix with distortion correction applied. As explained in Figure 3.3, the projection process excludes any error introduced in triangulation, the results can therefore be used as an indicator of the accuracy of the camera parameters. The pixel coordinates of the projected corners (yellow markers in Figure 3.4) were between 0.5 and 3 pixels out from the fiducial corners. Two issues became apparent while this test was carried out. Firstly, the SL reconstructed corners were identified manually from a point cloud (refer to Figure 3.1), even though the method was simple, it was still a subjective procedure as usually there were several points in close proximity of the central region, any of which could be the true corner. Secondly, the corner detector calculates pixel coordinates to 1 decimal place whereas in the actual reconstruction of the point cloud using structured light, projected corners all had integer pixel coordinates. This can contribute up to a 0.5 pixel difference.

The same projection test was also performed on the 3D coordinates recovered from extrinsic calibration. For these coordinates, the problem of choosing the right points from the point cloud was eliminated because each of them was originally derived from the pixel location of a fiducial corner. The task was therefore simplified down to the mapping the 3D coordinates onto appropriate pixel positions using equation 2.4. Despite eliminating the two possible sources of error stated in the previous paragraph, the discrepancies from the fiducial corners were still between 0.6 to 2.3 pixels (see Figure 3.5) which corresponded to a physical distance of around 0.5 to 2.0 mm on the board. An explanation for this was the discrepancies between the algebraic forward mapping (i.e. taking a photo of the scene) of 3D camera coordinates onto 2D pixel coordinates, and the numerical inverse mapping (i.e. during camera extrinsic calibration/tracking) from 2D pixel coordinates to 3D camera coordinates. If a camera model free of distortion was used, then the inverse mapping would just be solving for the camera coordinates (i.e. transformation matrix applied to the world coordinates X, Y, and Z) by taking the inverse of the intrinsic matrix A in equation 2.4. In that situation both forward and inverse mapping could be done by an algebraic expression. However with a high degree distortion correction in the camera model [34], the inverse mapping could no longer be described by a simple algebraic expression. In the Camera Calibration Toolbox [34], a numerical model was implemented for extrinsic calibration for this reason.

Apart from errors in the inverse mapping, error in the construction of the board, for example distance between two of the fiducial markers being 299.3 mm instead of 299 mm would also alter the accuracy of the projection. A number of tests on these were done, by intentionally altering the grid spacing in the software for amounts representative of the likely magnitude of errors (<1 mm), the effect was relatively minor in comparison with a less than 0.3 pixel offset for all the values tested.

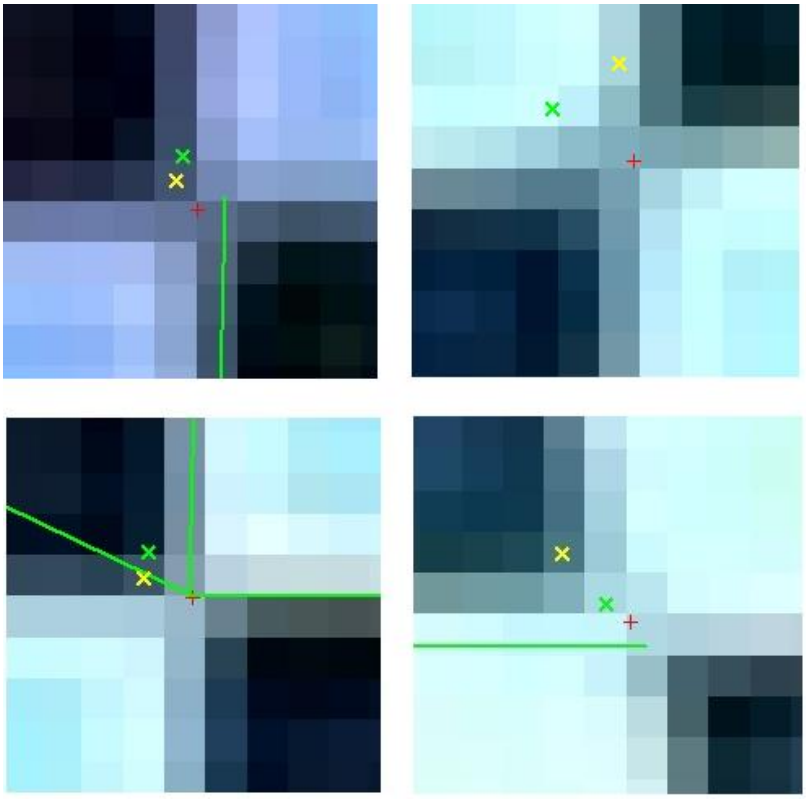


Figure 3.4. Accuracy of intrinsic camera calibration, showing the pixel coordinates of: 1. corners found by the corner detector (Red); 2. the coordinates in red transformed into 3D then re-projected back onto the image (Green); 3. 3D coordinates from SL projected onto the image (Yellow).

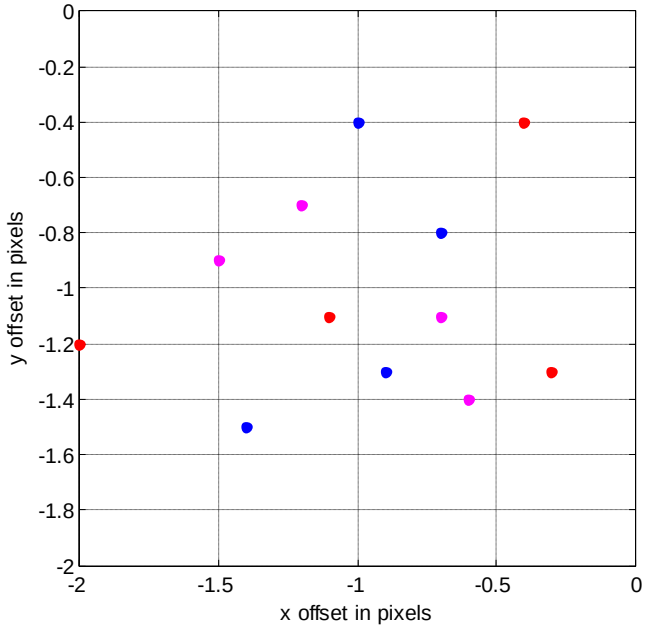


Figure 3.5. Discrepancies in pixel coordinates of fiducial corners and of the projected 3D coordinates.

3.1.3. Estimating Camera Tracking Inaccuracies

In the previous section, discrepancies were observed when the 3D coordinates recovered by extrinsic calibration were projected (forward mapped) back onto the original image. Pixel coordinates of the re-projected corners were all shifted towards the origin of the image coordinate system. The magnitude of the shift ranged from 0.3 to 2 pixels in the *x*-direction and 0.4 to 1.5 pixels in the *y*-direction (see Figure 3.5). The mean offset in *x*-direction was -1 pixel with a standard deviation of 0.5, in the *y*-direction the mean was also -1 pixel with a standard deviation of 0.4 pixels. The pixel distance between each pair of fiducial markers in the images ranged between 307 and 381. Based on this information, pixel length was approximated as 0.88 ± 0.10 mm. Using the estimated pixel length and the mean pixel offsets, a physical offset of +0.88 mm in the both *x* and *y* directions was applied to compensate for the observed shift. The errors in the 3D coordinates after this correction should be ± 0.60 mm in the *x*-direction and ± 0.50 mm in the *y*-direction.

The error in the *z*-direction was estimated by measuring the pixel lengths of the re-projected sides. Unlike in structured light reconstructions (refer to Table 3.2), the length between 3D corner coordinates recovered by extrinsic calibration would always be 299 mm (it is an input parameter in the extrinsic calibration). This meant the side lengths (in pixels) between the re-projected corners could only be in agreement with side lengths of the fiducial markers if the calculated distances to camera were also in agreement with the true distances to camera. Figure 3.3 can help visualize this. If a new grid, with identical dimensions to the largest grid seen in Figure 3.3 is added but at a closer distance from the camera, the projection of this new grid onto the camera would be larger than those of the original six. Conversely if the new grid was positioned further away, the projection would appear smaller. While the *z*-direction distance and the total distance between the camera and a point on the board were not identical, it was considered a reasonable approximation because all the coordinates on the board had a dominating z-component (see Figure 3.2).

3.2 Characterizing SL 3D reconstruction

In section 3.1, it was found through re-projection of the fiducial corners that the extrinsic calibration technique for finding the 3D position of the board corners was more accurate, contributing only a sub-millimeter error in each of x and y directions. Experiments described in this section were based on that finding, using dimensions

and coordinates tracked by the camera extrinsic calibration as the "true values" to assess errors in the structured light method pixel by pixel.

The 3D reconstruction of the plane was compared to a set of software generated surface coordinates derived from the camera extrinsic calibration. The transformation could be used to calculate the position in camera coordinates of any point on the plane. There were three different ways of generating the surface coordinates: (1) create a mesh grid of regular *x* and *y* values, forming an evenly distributed point cloud; (2) a grid with irregularly distributed points, only the *x* and *y* pairs that features in the 3D reconstruction are included; (3) in addition to (2), each point in the generated point cloud also has an unique correspondence to a point in the 3D reconstruction and vice versa. The correspondence is established between two points, if they share the same pixel position in the image coordinate system. Of the three methods, the third was deemed the most suitable for the purpose of this experiment because it allowed offset between the AR and the 3D reconstruction to be quantified for every encoded pixel in the image, which was a more realistic comparison than calculating perpendicular distances between points and the fitted plane or comparing only the *z*-component. Perpendicular distances can exaggerate the accuracy of matching, for example, if the reconstructed point cloud was known to be out by 3 cm in the positive *x* direction but the majority of the points still overlapped the AR surface, such a calculation would still conclude that the majority of the points in the reconstruction matched the AR surface when they in fact did not match.

The goal of this experiment was to characterize the differences between the structured light acquired surface of the calibration board, and the surface generated based on the camera registration of the calibration board.

3.2.1. Method

As in the experiments in 3.1, the board was scanned with the structured light sequence and the all-on image was used for extrinsic calibration. Extrinsic calibration results were again used to recover 3D coordinates of the board, but instead of just calculating the four corners, coordinates were calculated for every successfully encoded pixel within a rectangular region bounded by the four fiducial corners. This meant every encoded pixel within that region had two 3D coordinates, one calculated from the extrinsic calibration results, and the other from 3D vision using structured light.

The distances between the camera and different regions of the board were not identical, and because of this, the structured light reconstructed point cloud had a higher density of points in regions closer to the camera than regions further away. The effect can be recreated by a perspective transformation and the concept is explained in Figure 3.6. Imagine a camera capturing the image of a square positioned on a steep angle to the optical axis of the camera (Figure 3.6, left), while the image appears to have uniform resolution (shown as blue dots), the distribution of image pixels across the plane is in fact uneven and this can be seen by transforming the viewing direction to align with the normal of the plane (Figure 3.6, right).

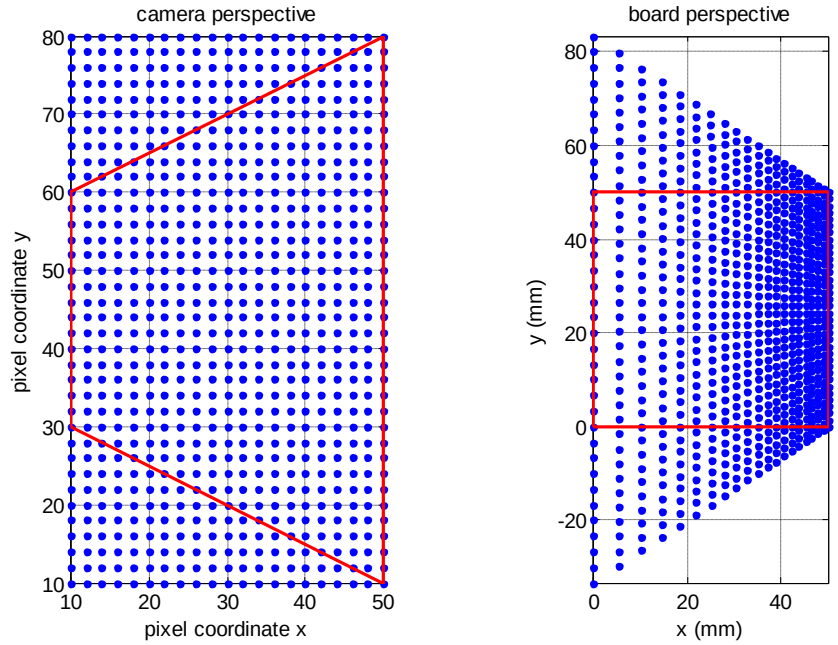


Figure 3.6. Left, image plane of a camera looking from a steep angle, at a square region bounded by the red lines, the blue dots represent camera pixels. Right, the same setup but now from the square board's perspective. This is an exaggerated example, in actual measurements, the camera look angle was never as steep as what is shown here.

The location of the pixels in physical coordinates could then be calculated using the perspective transformation, and based on the dimension of the square (Figure 3.6 right and Figure 3.7 left). From the extrinsic calibration results, the point cloud can be transformed (see Figure 3.7 right) by the exact same method as the transformation of the four fiducial corners in the previous section. The transformed point cloud is stored in an n by 3 array in which the n^{th} row represents the 3D coordinate of the same pixel on the image, as the n^{th} row structured light reconstructed point cloud (see Figure 3.8).

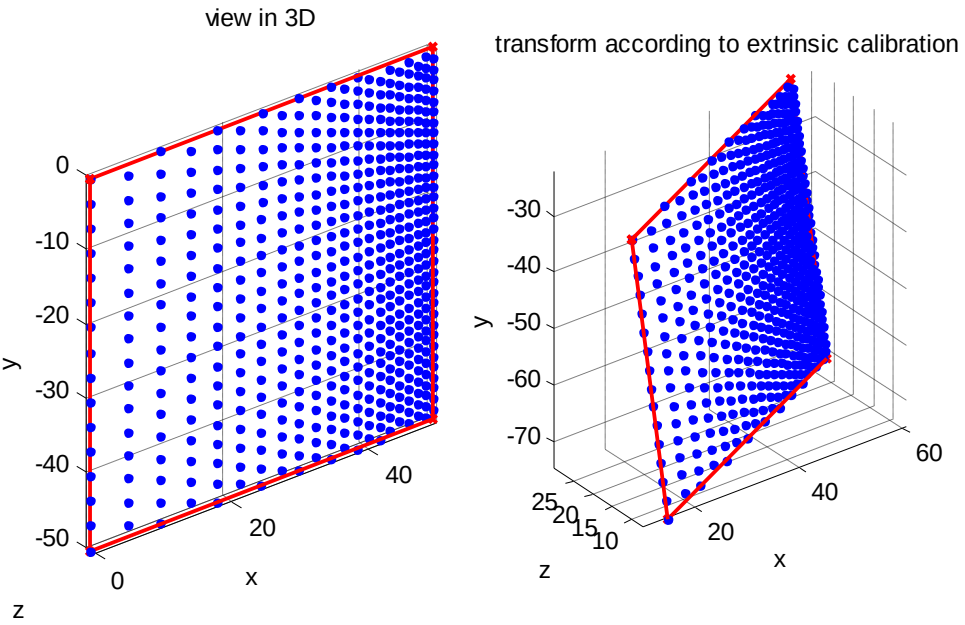


Figure 3.7. Recovering 3D coordinates for every image pixel within the square region using extrinsic calibration results.

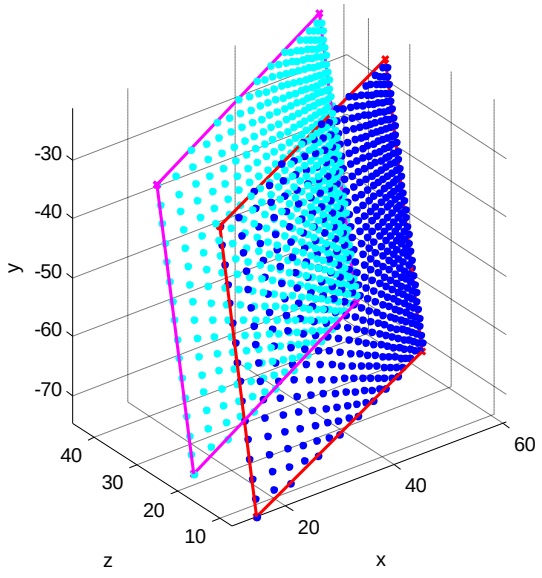


Figure 3.8. Every camera pixel within the red region (see Figure 3.6 Left) has its 3D coordinates calculated twice, one using structured light (cyan) the other using extrinsic calibration (blue). The point clouds shown here were generated for explanation purposes.

3.2.2. Results

Effects of triangulation angle

The structured light reconstructions of the plane presented in section 3.1 were re-analyzed using the described method. Offsets between the two types of measurement were calculated for each encoded pixel. For each pixel, the triangulation angle, which refers to the angle between a projector plane and an intersecting camera ray, was calculated and compared to the error at that pixel. They were observed to be inversely correlated in two of the three orientations (see Figure 3.9), with linear correlation coefficients of -0.78 (magenta) and -0.61 (red), and statistically independent in the third orientation with a correlation coefficient of 0.21 (blue).

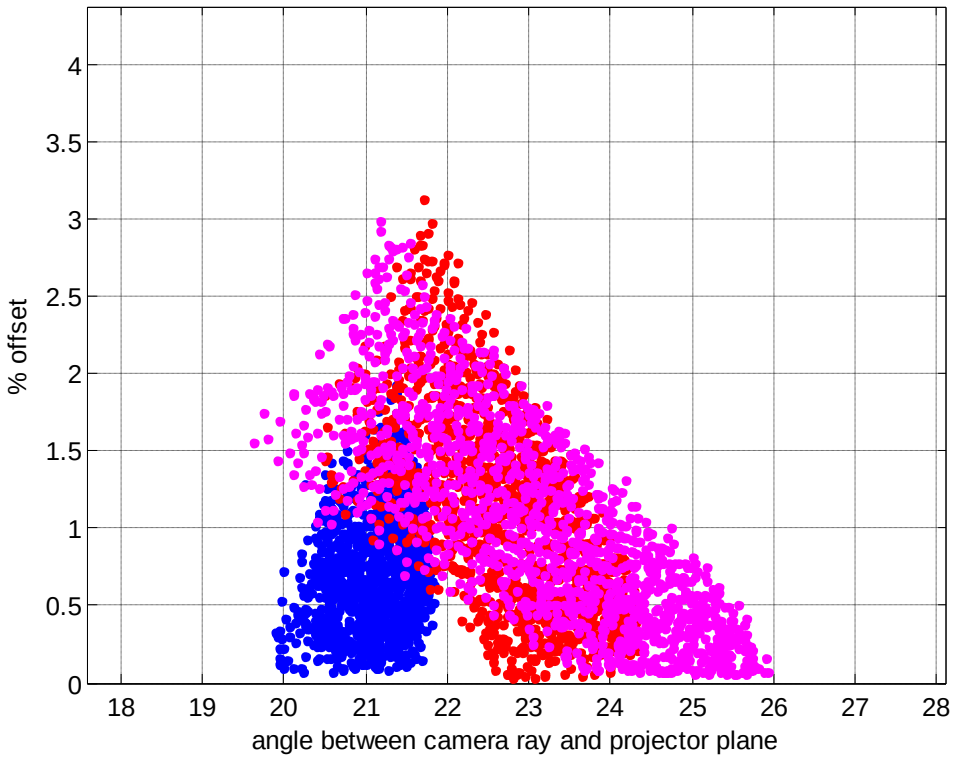


Figure 3.9. Relationship between triangulation angle and error.

An inverse correlation between triangulation angle and error is plausible. Consider a simplified triangulation problem in a 2D projector and camera system, let the angle between the projector ray and the projector optical axis carry an offset of -0.5^{o} (assume this came from inaccurate projector calibration), and the camera ray has a negligible error. If the true triangulation angle was 22^{o}, and the true point of

intersection was 1m from the projector (*l* = 1, see Figure 3.10) the offset of -0.5° will result in an error of 2.4cm over the distance of 1m, if the true triangulation angle was 26°, the same offset in angle would now generate a 2.0cm error, if the triangulation angle was increased further to 44°, the error would be lowered to 1.3cm. The general expression for the expected distance offset, *d*, is given by the sine rule:

$$\frac{d}{\sin D} = \frac{\mathrm{s}}{\sin \theta} = \frac{l}{\sin \varphi} \qquad (3.1)$$

where *l* is the length of the true projector ray vector **l**, θ is the true angle of triangulation, *s* is the length of projector ray vector **s** found by calibration, φ is the triangulation angle measured in calibration, and *D* is the angle offset. This explanation could not account for the large magnitude of error observed for the given range of triangulation angles, it never the less showed the accuracy of a structured light system is susceptible to the positions and view angles of the camera and projector with respect to each other.

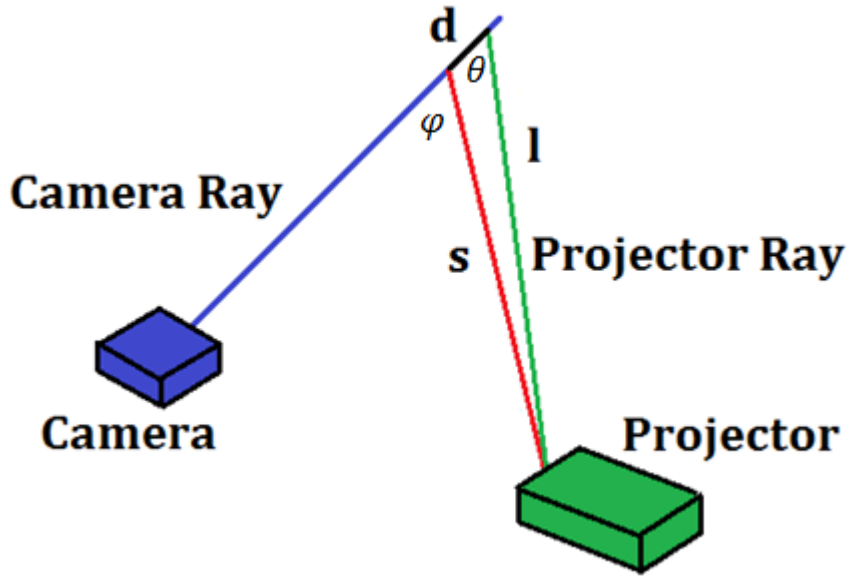


Figure 3.10. Error in the triangulation result (**d**) caused by a discrepancy between the true direction (green) and the calibration result (red) of a projector ray.

Accuracy across the projection field

Triangulation error and the encoding plane index at each pixel were also compared. The encoding plane index for a camera pixel refers to the column number on the 768 row, by 1024 column projection field, where the incident light originated from. The columns are numbered from 1 to 1024 starting on the far left edge of the projected light field. A stronger inverse correlation was observed in all the reconstructed planes

(see Figure 3.11) with correlation coefficients of: -0.8575 (blue), -0.9153 (red), and -0.9401 (magenta). For this particular projector and camera setup, the structured light reconstructions were less accurate in regions encoded by the left of the projection field where the system tended to overestimate the distance from the camera. The accuracy of the reconstruction improves linearly towards the centre of the projection field. The right part of the projection (columns 650-1024) was not used due to the location of the board during encoding.

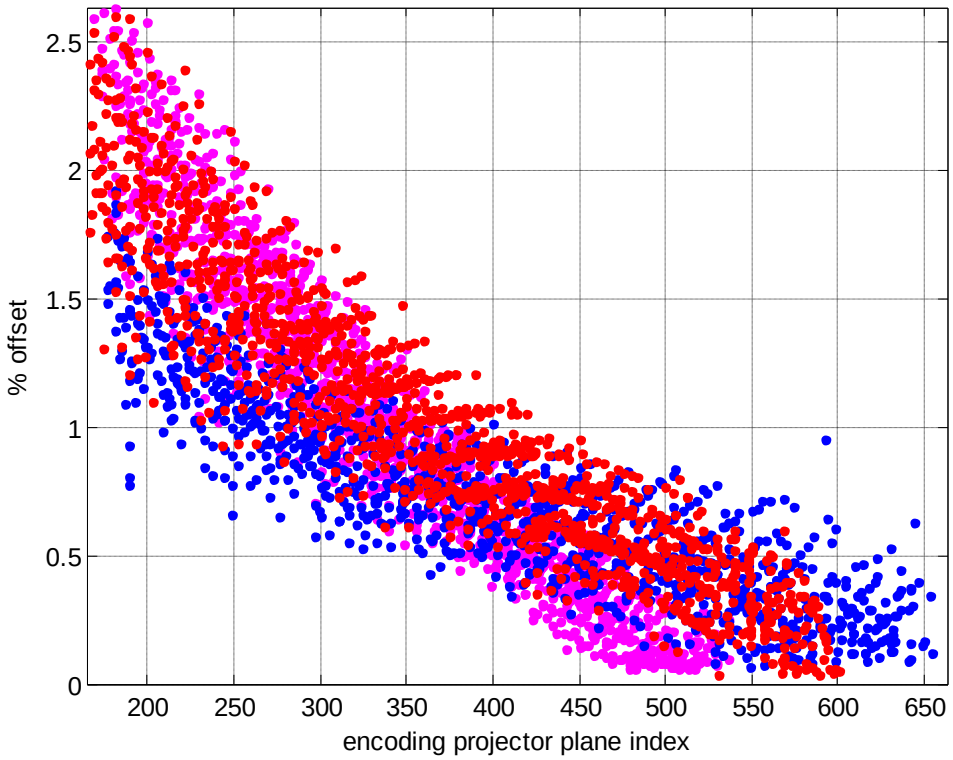


Figure 3.11. Relationship between the encoding plane indices and error.

Angle between projector planes

The angles of separation between each pair of neighbouring projection planes were calculated. This was done to help visualize the orientations of the projector columns in space, which was used as a check for the validity of the projector calibration result. The angle profile across the width of the projection field took the shape of a parabola (see Figure 3.12), it peaked at 0.0361^{o} around column 489 of the projection, and fell to around 0.0316^{o}, and 0.0309^{o} towards the left and right sides respectively. The varying separation angle of the projecting columns was not unexpected, digital projectors were designed for use on a flat screen , the distance light traverses from

the projector to the edge of the screen is greater than the distance required to reach the centre of the screen. This simple geometric effect causes the projection to enlarge towards the edges. While no supporting evidence could be cited, it was assumed that the gradual decrease in separation angle between neighbouring columns or rows, from the centre towards the outside of the projection field, was implemented in the projector design to compensate for this geometric effect.

The vertex of the parabolic separation angle profile was located around the 489th column, this was in accordance with the x-component of the principal point calculation which indicated the projector optical axis crossed its image plane at x = 489.4 and y = 791. While planes formed by the projector rows were not used for triangulation, it was still interesting to note that the y component of the principal point was not on the image plane. This was why a warning message stating the principal point could not be estimated was shown with the above result. Here lies a key difference between a camera and a projector: the image plane of a projector very rarely aligns with its optical axis. By design, there is usually a big vertical shift of the image plane away from the optical axis so a projector sitting flat on a tabletop can display an image onto a screen higher than itself.

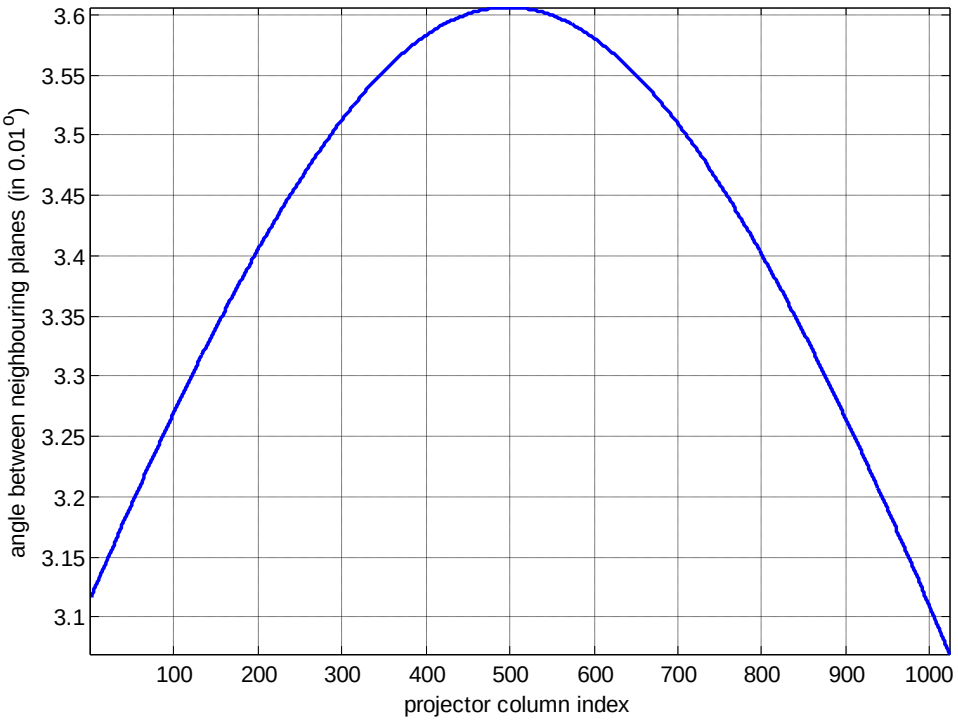


Figure 3.12. Angles in 10^{-2} degree between pairs of neighbouring, vertical projector planes.

3.3 Improved setup

In sections 3.1 and 3.2, some factors hindering the accuracy of the structured light reconstruction were identified. These were:

1. Inverse mapping in the camera extrinsic calibration - this error first became apparent when 3D coordinates of the fiducial markers were re-projected back onto the original image. This error source would most likely have affected the structured light reconstruction as well, because one of the key steps in the calibration of the projector involved camera extrinsic calibration (refer to section 2.3.2).
2. Inaccurate point of intersection in triangulation - in section 3.2.2 one reason given for a possible inverse correlation between triangulation angle and error was that small triangulation angles magnify the error. While this reason alone could not explain the observation, increasing the triangulation angles should in principle improve the accuracy of the reconstruction.
3. Other factors - in section 3.2.2, varying inaccuracy across the width of the reconstructed board was observed, but no obvious explanation for this observation could be found.

A set of new measurements were taken with modifications made to reduce the effects of factors 1 and 2. It was hoped that by reducing their effects, other sources of error would become apparent.

3.3.1. Improving Calibration Accuracy

The re-projection errors on all camera calibration images were checked using the "Analyse error" tool in the camera calibration toolbox. This function calculates the pixel error of the re-projected grid corners relative to their initial image coordinates found by the corner detector (See Figure 3.13). The calibration images that gave large re-projection errors, for example those shown in blue, green, and magenta in Figure 3.13, were identified. The corner finder script was run to re-compute image coordinates of every grid corner in these images using a bigger search window centered on the existing image coordinates. The method was originally used by the author of the calibration toolbox in the first calibration example presented on his website. In that example, he was able to reduce re-projection errors from the initial 0.45 and 0.39 pixels (*x*, and *y* direction respectively) down to just over 0.1 pixels in both *x* and *y*.

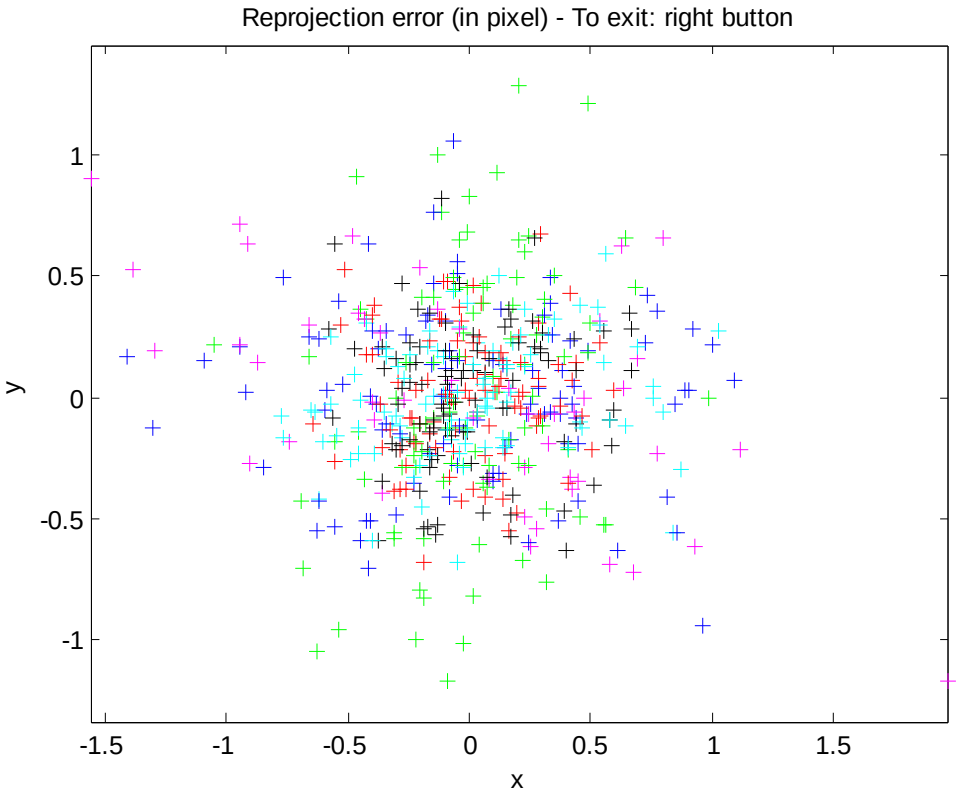


Figure 3.13. Plot showing re-projection errors (in pixels) in the camera calibration images. The data points were colour coded based on their image numbers. An accurate calibration would yield a tight cluster of points around (0,0) on this plot.

The mean pixel error before any corrections were made was 0.41 and 0.36 pixels in the x and y directions respectively. Out of the 11 calibration images, three were considered inaccurate with multiple error points going outside a 2 pixel by 2 pixel box centered around (0, 0), and only two were considered accurate with more than 80% of error points lying within a 1 by 1 pixel box. The grid corners on all 11 images were re-computed using a 25 by 25 pixel window as opposed to the 11 by 11 the calibration toolbox had by default. The camera calibration routine was run with the new grid corner coordinates, and re-projection error in the new camera parameters analyzed (see Figure 3.14). The larger search window size improved the mean re-projection error to 0.38 and 0.31 pixels in *x* and *y* directions respectively. Window sizes larger than 25 by 25 were also tested, but they did not reduce the error any further. The reduction in mean re-projection error meant the new set of camera parameters gave a more accurate description of how the camera mapped 3D coordinates onto a 2D image than the previous set of parameters.

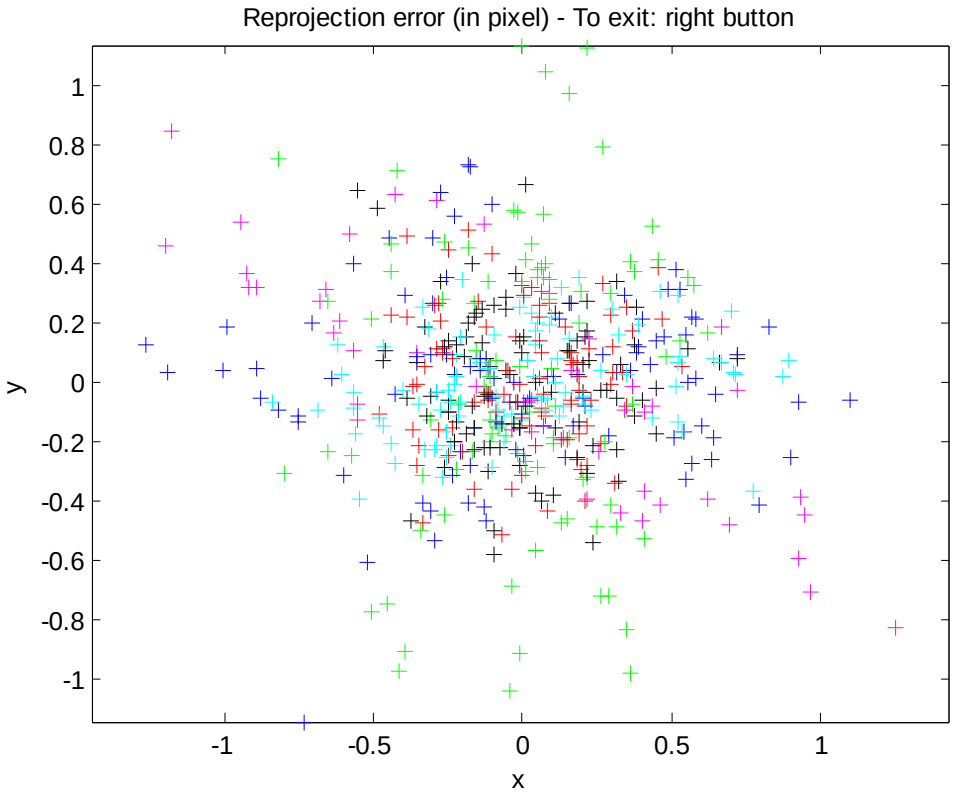


Figure 3.14. Re-projection error (in pixels) of grid corners found using a 25 by 25 search window in the camera calibration images

3.3.2. Camera/Projector positioning

In section 3.2.2 on the effects of triangulation angle, it was shown by simple geometry, that a larger triangulation angle could reduce the effect of inaccuracies in projector plane calculations. Larger angles can be achieved by rotating either device towards the optical axis of the other. Doing so however, would make the field of view for the overall structured light system (e.g. the shaded regions in Figure 3.15) come closer towards the devices, and hence make the encoding pattern projected onto the scene to go out of focus. The solution to this was to also increase the separation between the camera and the projector (see Figure 3.15)

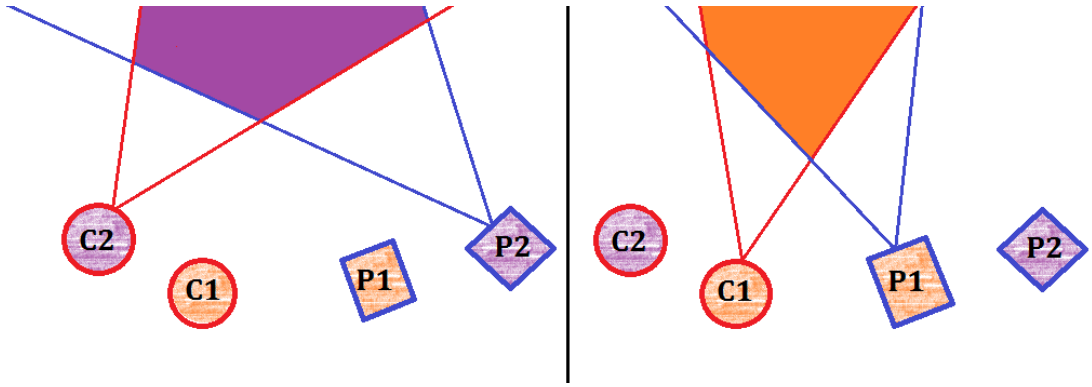


Figure 3.15. Schematic drawing comparing large triangulation angles (Left) and smaller triangulation angles (Right). C1 and P1 represent the locations of the camera and projector respectively in the original setup, C2 and P2 represent their respective locations in the new setup. Red and blue straight lines represent the boundaries to their respective field of view.

3.3.3. Results

The camera and projector positions were adjusted to increase the average triangulation angle from 23^{o} to 40^{o}. The system was recalibrated twice, with and without the re-projection error analyses described in section 3.3.1. 3D images of the board were captured at five different rotation angles about the *y*-axis (-30^{o}, -20^{o}, 0^{o}, 20^{o}, 30^{o}), at each orientation, two 3D surfaces of the board were reconstructed one for each set of calibration results.

The new camera-projector setup reduced the mean differences between the structured light reconstruction and camera tracking based on extrinsic calibration. The mean percentage offsets were under 0.7% for all five orientations, down from 1.2% in the previous setup. The amounts of offset in each spatial dimension were all in agreement amongst the five orientations. The best agreement between the two measuring techniques was in the *y*-dimension where the results differed by less than 0.15%. In the *x* and *z* dimensions, the consistency of the offset measured across the range of rotation angles meant that correction values could be numerically derived to compensate the discrepancies between the measuring methods. The structured light system underestimated the distance of the board from the camera by 3.07 to 5.57 mm (see *z*-offset in Table) and a lateral shift of between 3.61 and 4.15 mm in the positive *x*-direction. A correction factor of 4 mm in the *z*-direction and 3.8 mm in the *x*-direction could be used to reduce the observed discrepancies between the two measured surfaces. With the correction factors applied to the *x* and *z* components of the structured light reconstruction, the offset between the two techniques was

further reduced to ±1 mm, ±1.5 mm, and ±4 mm in *x*, *y*, and *z*-directions respectively. The corrected offsets between the two measuring techniques can be used as the error of the overall system. In the context of a radiotherapy patient setup, this would mean any misalignments detected by the system with magnitudes less than 1 mm, 1.5 mm, and 4 mm in *x*, *y*, and *z* components could simply be "false positives" rather than actual alignment errors.

y-axis rotation	Sample size	x-offset (mm)	y-offset (mm)	z-offset (mm)	Length of sides (mm)
-30^{o}	86224	4.15±0.50	0.011±1.17	3.43±2.03	296.5±1.9
-20^{o}	97940	3.89±0.55	0.067±1.12	3.73±1.95	296.6±1.1
0^{o}	109224	3.61±0.65	0.100±1.09	4.48±1.75	297.0±1.1
20^{o}	100244	3.97±0.38	0.120±1.09	3.07±2.02	297.8±1.0
30^{o}	95871	3.85±0.46	-0.370±1.26	5.57±2.57	298.0±0.6

Table 3.3. Directional offsets in the coordinates of the calibration board recovered from (1) camera tracking of the board, and (2) structured light reconstruction of the board.

The correction factors were derived from measurements of the board within the range of 900 mm to 1100 mm from the camera, and should therefore only be applied to objects within the same range of distances. The reason for not experimenting with a bigger range of camera-to-object distances was related to the field of view of the camera-projector system and the size of the board. The field of view (FOV) of the 3D vision system was considered as the region in space where the features on the encoding patterns were easily identifiable and each encoding pattern was visible in its entirety on the camera. FOV of the vision system is therefore the combination of the camera's FOV and the projector's range on the given lens setting. If the board was moved too far away from the camera, the projected encoding pattern became out of focus thus hindering the encoding accuracy. Adjusting the lens on the projector was not an option because doing so would modify the intrinsic parameters of the projector. On the other hand, due to the size of the calibration board, moving it too close to the camera could make the fiducial markers go out of the field of view, making recovery of the board position through extrinsic camera calibration impossible.

Re-projection errors in the calibration images were analyzed and reduced using the method described in 3.3.2, the updated camera-projector system calibration parameters were used. The re-calculation of grid corners on calibration images,

however, seemed to have enhanced the discrepancies between the two measuring techniques (compare Table 3.3 and Table 3.4). Reconstruction using the new calibration parameters yielded differences of over 24 mm between extrinsic tracking and structured light reconstruction. The disagreement was most apparent in the *z*-direction (see Table 3.4). The calibration was repeated but it did not alter the result significantly. It was possible that the discrepancy shown in Table 3.4 was a better depiction of the capability of the current method, and judging by the magnitude of the discrepancy, there was likely another major source of error unaccounted for. It would also mean the outliers seen on the re-projection plot reduced the effect of this error source, perhaps coincidentally.

y-axis rotation	Sample size	x-offset (mm)	y-offset (mm)	z-offset (mm)	Length of sides (mm)
-30^{o}	86224	0.692±2.13	-2.43±2.67	26.2±1.07	293.9±2.6
-20^{o}	97940	0.254±2.52	-2.46±2.65	26.8±1.23	293.6±3.2
0^{o}	109224	0.375±2.94	-2.52±2.59	27.2±2.09	292.8±2.1
20^{o}	100244	2.71±2.08	-2.49±2.30	23.0±1.61	292.7±0.67
30^{o}	95871	2.78±2.13	-3.05±2.50	25.3±1.29	292.5±0.94

Table 3.4. Directional offsets after corrections were made to the pixel-positions of the grid corners on the calibration patterns.

Chapter 4

Visualization and Full System Setup

The temporal structured light method was incorporated into a simplified AR, where the AR would be used for initial qualitative alignment. The surface was then reconstructed to provide a quantitative account of alignment accuracy. The reconstruction was rendered using a colour scheme representative of the offset between the measured surface and the planned location of the surface.

4.1. Visualization

The 3D reconstruction technique produced a point cloud which was compared to the CT reconstruction of the object to assess accuracy of alignment. Quantifying the exact magnitude of the discrepancies between the two reconstructions posed a technical challenge. In the measurements of the plane, both techniques used images captured by the same camera to generate their respective point cloud. Since the camera position and direction of its optical axis did not change between the two measurements, the correspondences between points on the two point clouds could be easily identified based on their pixel locations in the original camera image. With CT reconstructions this was not possible. There was no simple way of identifying a corresponding point in the CT data for a given point in the 3D vision captured point cloud.

To a certain extent, the registration problem is similar to the correspondence problem in stereovision. The key differences are the points in stereovision are 2D image coordinates, and the points are coloured which assists the matching process. The complexity in implementing a 3D feature matching algorithm is beyond the scope of this work. Another possibility was to make the surface matching into an optimization problem. This meant finding the rigid-transformation (three rotation angles, and *x*, *y*, and *z* translations) of the 3D vision reconstruction, that minimizes the cumulative nearest distance to the CT reconstruction. While the transformation parameters found could also be used to fine tune the couch position, unlike present rigid-transformation based procedures, its main purpose here would be for the

registration of the two surfaces for a more accurate quantification of surface deformations. The optimization option should be the more manageable route to take, but due to time constraints it was not implemented in this project. However, a follow up project using an existing software package to achieve this is currently in progress (see Chapter 6)

4.1.1. Description and Justification

The method implemented for visualizing offsets was a straight forward subtraction between the *z*-components of the CT and 3D vision point clouds. This means that while capable of generating a full 3D point cloud of the surface, the 3D vision system instead served more like a depth offset detector. In a stand-alone 3D vision system, this kind of comparison would be an over-simplification because no consideration was given to offsets in *x*, *y* directions, and the three rotation angles. However, since the 3D vision system was to be used in conjunction with Augmented Reality, in a controlled setup AR should already be able to provide good alignment in all but the dimension along the camera optical axis (see Figure 4.1). The use of such a simple subtraction could then be justified because it added to where AR was at its weakest. Talbot (2009) [19] offered a good example to demonstrate this. In his investigation into optimal camera placement, AR was set up in coronal (Figure 4.1 top) and sagittal views (Figure 4.1 middle) of a phantom flat on a table top. In the coronal view, alignment in the *x* and *y* directions could be achieved with high accuracy and offsets even on a pixel scale could be noticed visually (for example, near the shoulder of the misaligned arm). The coronal view, however, failed to allow any visualization of the significant *z*-direction offset which was partly revealed in the sagittal view. A single sagittal view in this example, could only offer *z*-direction offset visualization to, at most, 50% of the phantom surface.

Figure 4.1. AR based alignment of a phantom in Talbot (2009) [19]

If *z*-direction depth map acquired by the 3D vision system had been used in conjunction with the coronal view (see Figure 4.1, top), one view of the phantom could have yielded a more complete depth profile than using all three AR views of the existing system. It has to be re-iterated that the use of the method could only be justified when good alignment in the *x* and *y* directions had already been achieved through AR guidance.

4.1.2. Implementation of Method

The visualization was implemented in Matlab as described in the previous sections. Before comparisons could be made between CT and 3D vision data, both must be processed so they are in the same format. The CT data consisted of 198 DICOM files of an anatomical phantom (see Figure 4.2). These were opened in CERR [40], a software platform written in Matlab for display and analysis of radiotherapy treatment plans. In CERR, the external surface of the phantom was contoured from every slice of the CT reconstruction. This was done with the aid of a built in boundary detecting algorithm. The contoured CT slices were then exported from CERR as 198 new DICOM files, one of which was a DICOM RT structured file (usually named as RS*.dcm) that contained the contours for all 198 slices. These contours were extracted and their coordinates combined into one point cloud. The CT point cloud was then transformed into camera coordinates according to the extrinsic camera calibration.

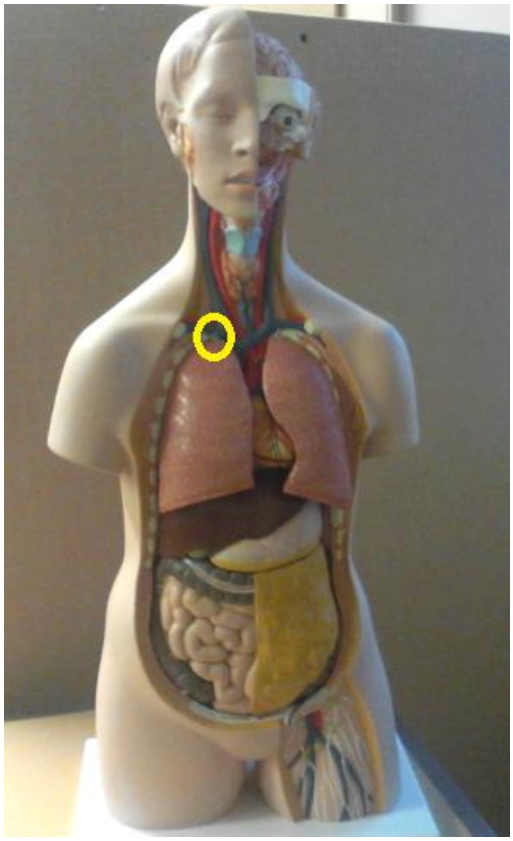


Figure 4.2. Anterior view of the anatomical phantom, one of the lung hooks was chosen as the point of reference (enclosed in yellow) to be aligned to the origin of the world coordinate system. It was chosen for its metallic composition that was easily identifiable in the CT reconstruction.

In 3D vision reconstruction, the background had to be removed. The number of points originating from the background could be minimized in the setup by using black absorbent materials in the background. The room used was a laser lab which had black curtains that worked very well as background. On an actual linac couch, the background would not be as customizable as it had been in the lab. An algorithm that determines whether a point belongs to the background based on distance from the camera could be easily added in these circumstances.

At this point the two point clouds could be compared visually (see Figure 4.3) but the *z*-component offset between the two could not be directly calculated yet due to differences in *x* and *y* intervals between the two data sets. The Matlab function *griddata* was used to interpolate both sets of data onto one common grid of *x* and *y* values. *Griddata* interpolates the *z*-components of a set of non-uniformly spaced vectors, such as the 3D vision point cloud, onto a user defined set of uniform *x*-*y* grids. This made it possible to calculate the differences between data sets in the *z*-component alone, and also enabled the use of Matlab's surface rendering functions which require the input array of vectors to be on a uniform mesh grid.

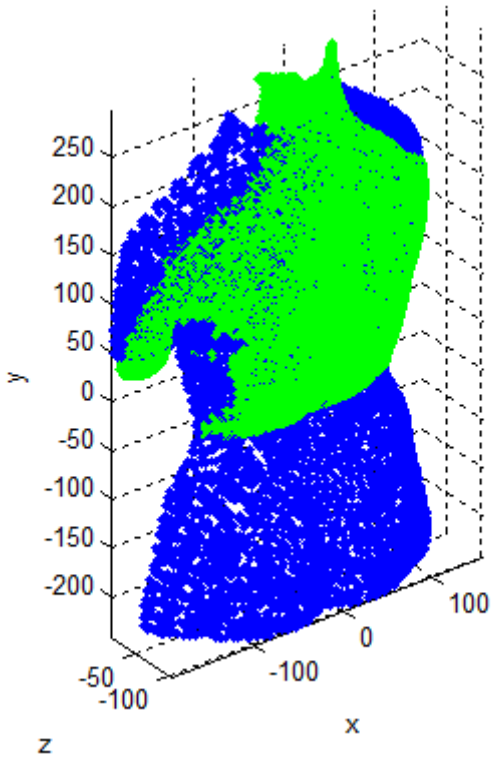


Figure 4.3. Visual comparison of the point cloud from CT reconstruction (blue) and the 3D vision point cloud (green).

Griddata uses a user selected method to interpolate the matrix for the *z*-values. The choices are [41]: triangle-based linear interpolation, triangle-based cubic interpolation, or nearest neighbour interpolation. One problem with *griddata* was that it would interpolate the surface point-clouds at *x-y* points outside the outer *x-y* boundaries of the original surfaces. Ideally, there should be a sharp cut-off to a constant *z*-value representing the background in the interpolated surface, at *x-y* points either sides of the original boundary. This could not be achieved with *griddata* alone. Figure 4.4 is a rendered surface of the green point cloud in Figure 4.3 (the colour scheme is explainedlater in this section). The visual artifacts caused by *griddata* at *x-y* locations outside the original boundary are very prominent.

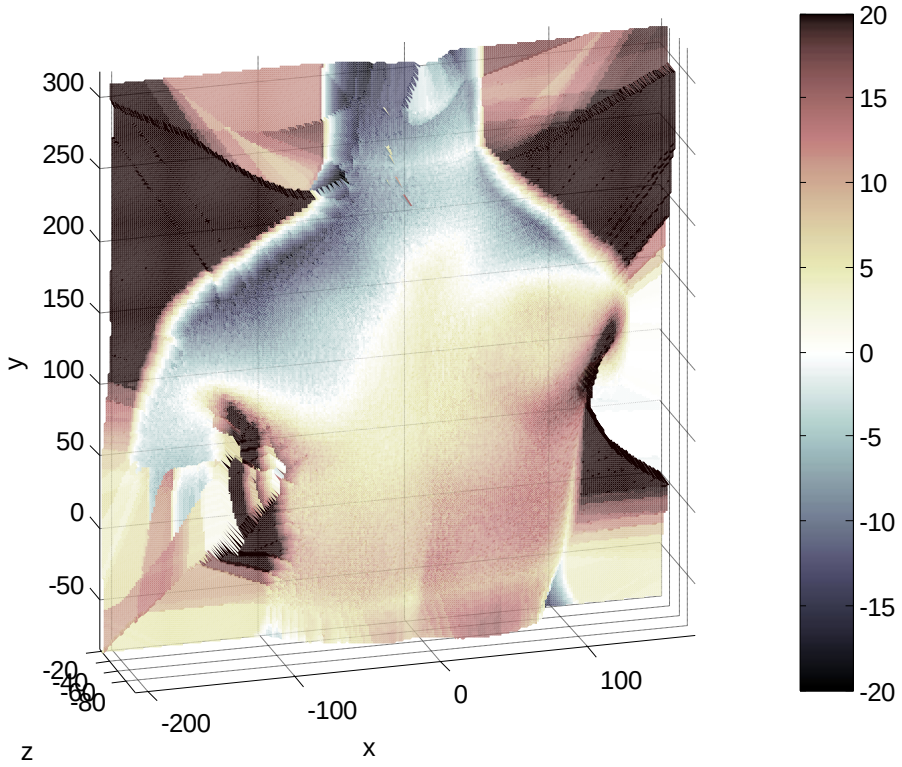


Figure 4.4. The structured light captured point cloud interpolated over a uniform x-y grid. The colour rendering was based on the *z*-direction offset with the CT point cloud interpolated at the same *x-y* grid locations.

A mask was implemented to eliminate the artifacts caused by interpolating the surface beyond the original boundary. The mask was a matrix identical in size to the uniform *x-y* grid where *griddata* interpolated points at. For every given matrix index, the mask had a value zero if the *x-y* coordinate of the grid at that matrix index was outside the boundary. The Matlab function *bwboundaries* was used to create the mask. *bwboundaries* is a boundary tracing tool for digital images in the image

processing toolbox. When used with two output arguments, for example: `[B,L] = bwboundaries(...)`, the second output L is a matrix, same number of columns and rows as the input image matrix, with nonnegative integer entries at locations that are part of contiguous regions, and zero-valued entries at locations that represent the background. The second output argument (L) of *bwboundaries* therefore would do exactly what the mask should (see Figure 4.5). Since *bwboundaries* was designed for images, and not a 3D point cloud, an image representation of the point cloud had to be created first. A zero matrix with the same dimensions as the mask was created. Matrix element $M_{a,b}$ was set to 1 if one or more points of the point cloud existed within the confine: $a - 0.5 < x < a + 0.5,\ b - 0.5 < y < b + 0.5$. This in effect produced a binary image that distinguished only the background from the surface and disregarded any variation on the surface itself. This was sufficient for the purpose of boundary tracing.

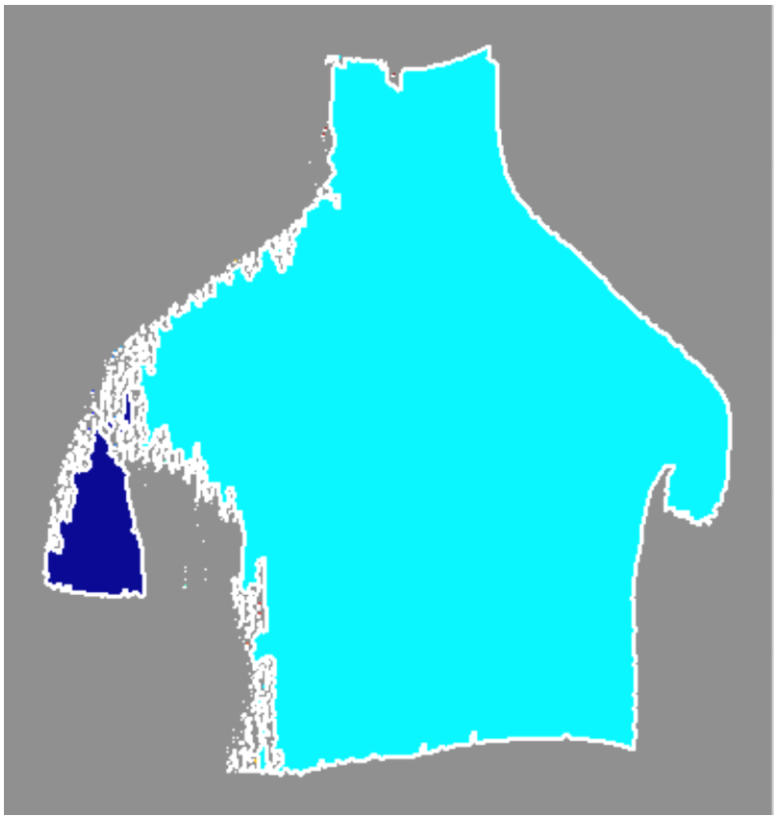


Figure 4.5. The second output argument of *bwboundaries* shown as an image. Grey represents the background, white represents all pixels immediately bordering the background, and the other colours represents contiguous regions.

Applying the mask as it was in Figure 4.5 still had one drawback, which was the large number of holes between contiguous regions that were too big and disconnected to be filled by the Matlab function *imfill*. These holes, particularly prominent around

the right shoulder region (refer to the left side of Figure 4.5), were caused by the lack of data points on that side of the reconstruction. The explanation for this was similar to that given for Figure 3.6 (refer to section 3.2.1). If this mask was applied to a rendered surface, it would create a large number of sharp "spikes" towards the background value, on the surface. To reduce this effect, a compromise was made by making the mask "fuzzier", which was done by setting any zero matrix elements immediately bordering one or more non-zero elements to one.

An additional function required of the mask was to help discarding regions of the two data sets that did not overlap. This was easily achieved by multiplying the mask for the CT data by that for the 3D vision data, element by element or by matrix multiplication of one by the transpose of the other. The last step in creating the mask was to set all nonnegative integers to one, and the final result can be visualized in Figure 4.6. Figure 4.7 shows the improvement made on Figure 4.4 because of the mask.

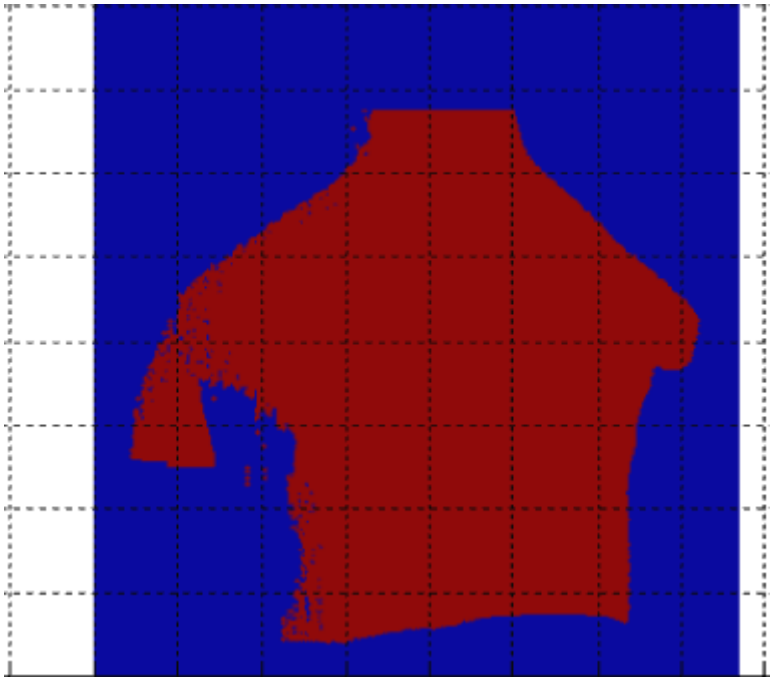


Figure 4.6. The final mask with red coloured pixels representing one-valued matrix elements and blue pixels representing zero-valued elements.

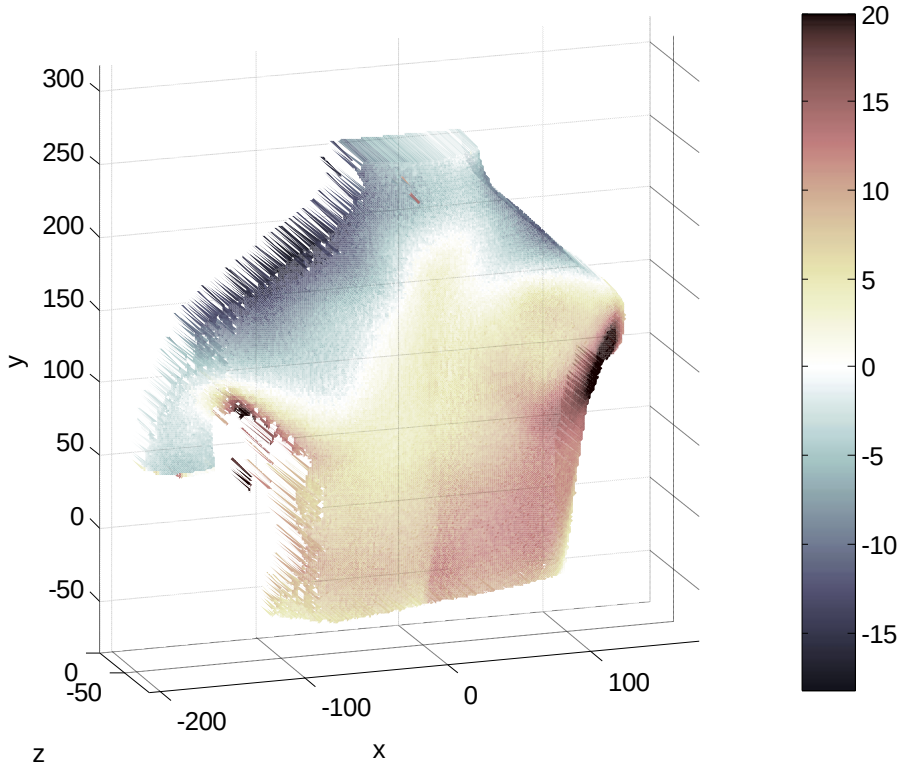
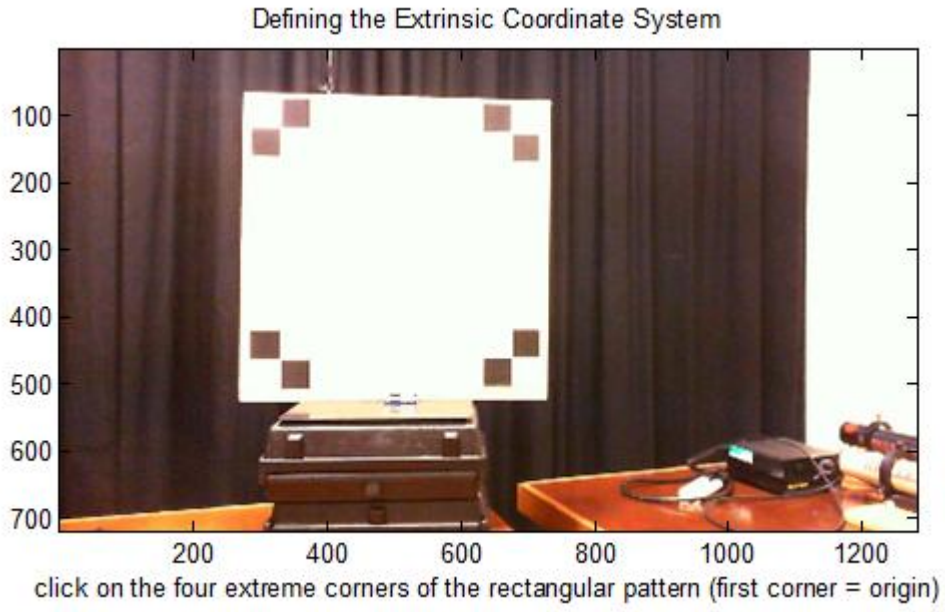


Figure 4.7. The same surface seen in Figure 4.4 with the mask applied. The mask set *z*-values of vectors beyond the original *x*-*y* boundary to zero, hence the *z*-difference calculation yielded zero (shown by the colour white) for all of these points.

4.2. Full System Setup

A simplified version of the AR system was implemented to demonstrate the concept of a 3D vision aided AR alignment system (see Figure 4.8). The AR program was implemented in Matlab to do the following. It begins by prompting the user to place the camera tracking device in the desired location then hit any key. The program takes an image of the scene and calls up the extrinsic calibration function, which determines the transformation parameters from the image. Based on the intrinsic camera parameters acquired in an earlier step and the transformation, the program calculates the distortion corrected projection of the CT point cloud onto the camera image plane. At the completion of the extrinsic calibration for AR, the user is asked to place and adjust the phantom in the scene, then press any key to have a snapshot of the scene taken. The captured image is displayed along with the AR virtual point cloud. The user is asked in the command prompt whether the alignment is satisfactory, the program loops back to the adjustment and snapshot step, until the user is satisfied with the alignment. The ideal AR system for alignment purposes

should display the virtual point cloud onto a live video feed. Unfortunately this could not be implemented in the full system due to an issue with the available hardware i.e. the webcam's auto-refocus function.



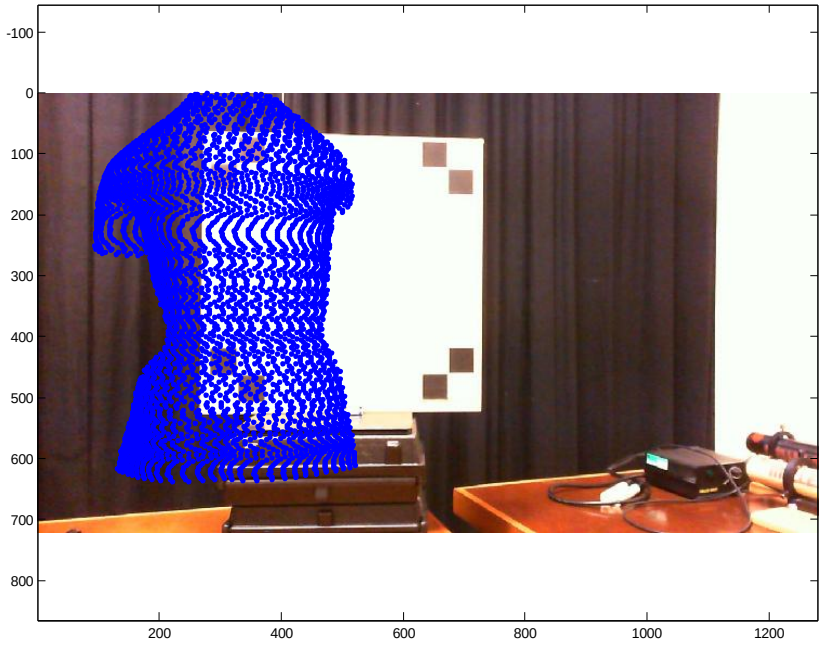
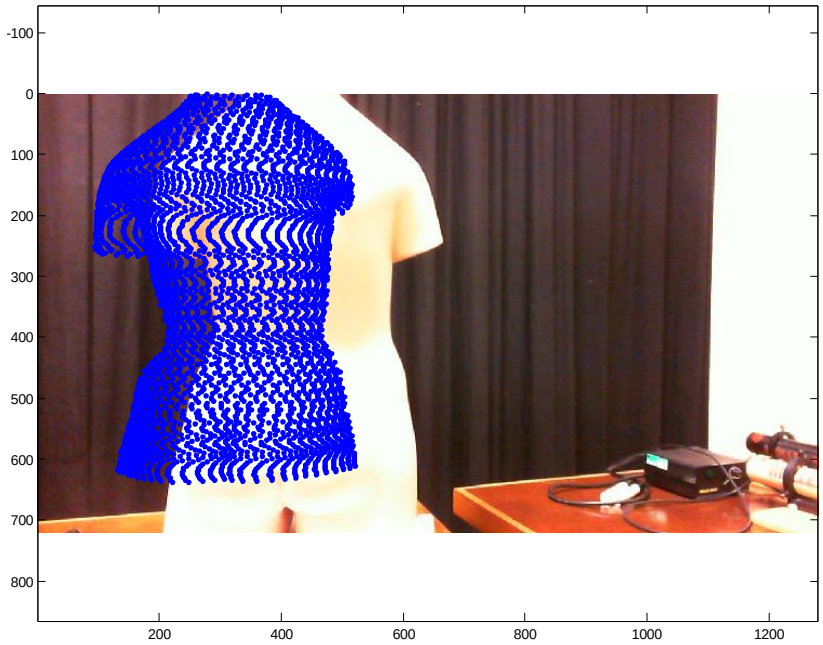
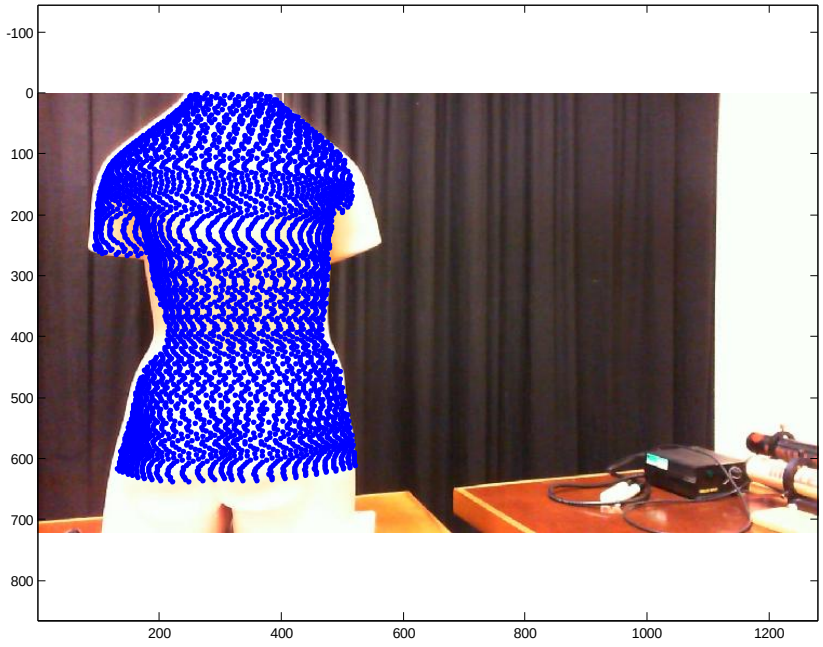


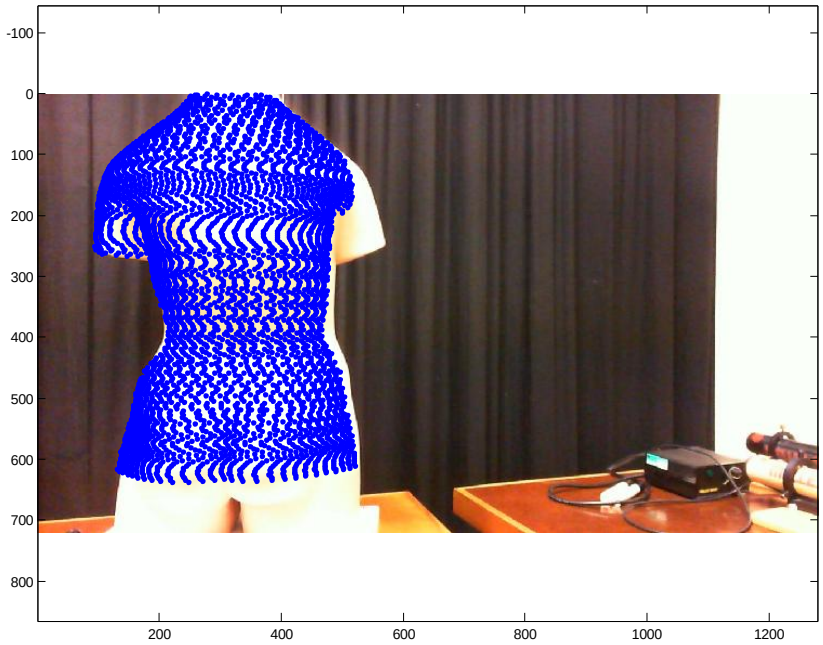
Figure 4.8. Top: camera tracking, where the fiducial on the top left corner marked the origin of the world coordinate system. Bottom: projecting the CT acquired point cloud onto the image.



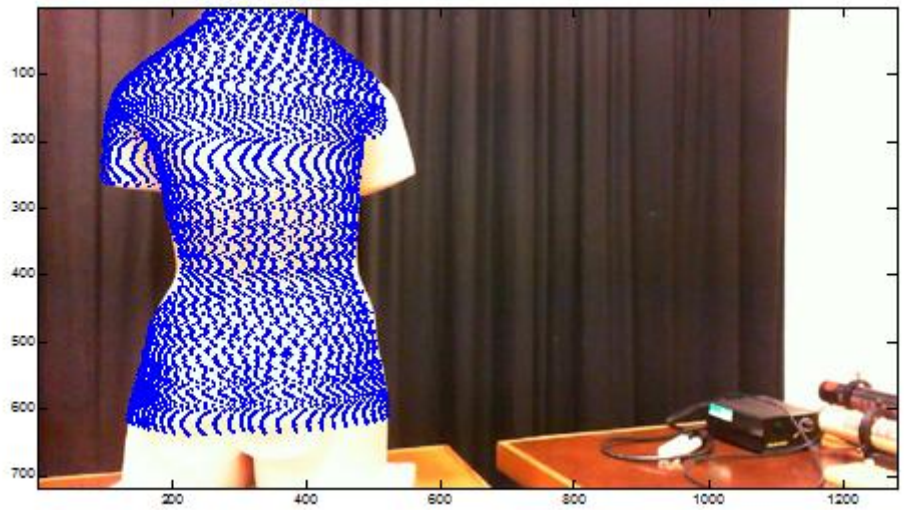
(a) The tracking device was removed and the phantom was placed in the scene, the initial placement was clearly too far to the right.



(b) The phantom seem to be in the correct *x* and *y* locations, but it was too close to the camera (see, for example, around the waist where the phantom appeared larger than it should be).



(c) The phantom now appears to be in the correct *x*, *y*, and *z* location, but the rotation about the *y* axis seems to be slightly off.



(d) Visually, this is a good alignment. A perfect alignment was not possible in this example due to the *y*-axis of the world coordinate system not being perpendicular to the table top, which was caused by the calibration board not standing at a full 90^{o} to the surface of the table.

Figure 4.9. Alignment with the aid of AR.

The structured light scripts were run after satisfactory visual alignment. The orientation of the phantom with respect to the projector created many blind-spots for the vision system. This was most prominent around the right shoulder and also the top of the right arm (see Figure 4.10) where large areas of the surface were in the shadow of other parts of the phantom, and thus could not be encoded by the projector.

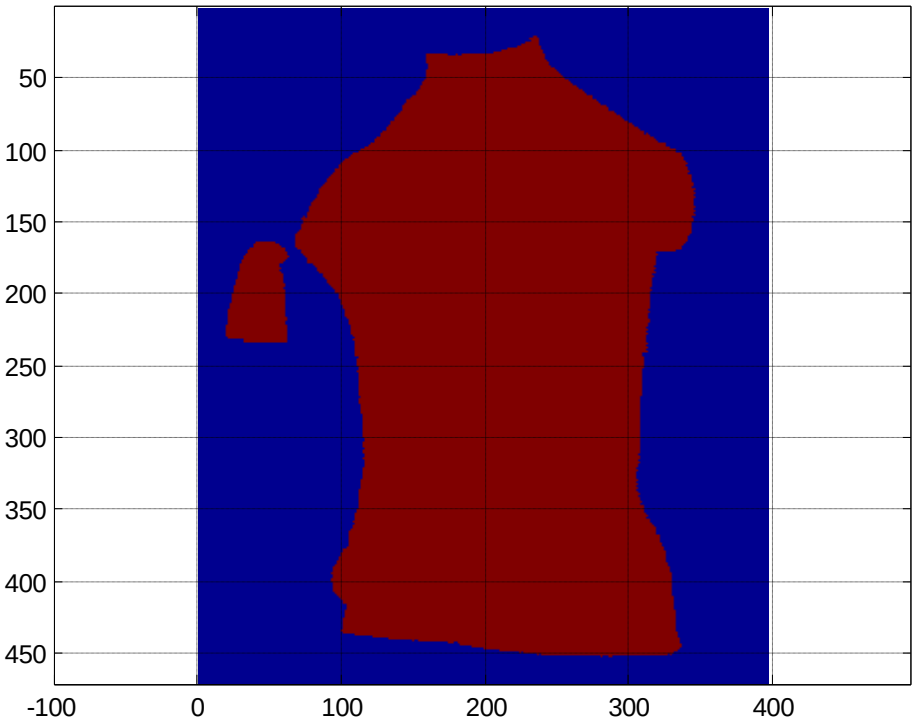


Figure 4.10. Mask for the setup seen on Figure 4.8. Regions where both CT data and 3D vision data were available are shown here in red.

The colour scheme for surface rendering was specifically designed for *z*-offset visualization. Ignoring any inaccuracies associated with camera tracking and in structured light reconstructions for now, then let the SL reconstruction be the actual position of the phantom, and the CT data the optimal position of the phantom. Depending on whether a region on the SL reconstruction was closer to, or further away from, the camera than its optimal *z*-location, it was visualized in a cold hot themed colour scheme. The colour-bar was rescaled based on the range of offset values between the surfaces. The rescale was always in such a manner that near zero offsets would be shown in white, and the maximum offset would appear black (refer to the colour-bars in Figures 4.11, 4.12, and 4.13 (b)).

The 3D structured light reconstruction of the phantom seen in Figure 4.9 (d) was rendered using the described colour scheme (see Figure 4.11). The reconstruction had a mean *z*-direction offset of -1.5±0.6 cm from the desired position. The measured offset was relatively uniform across the reconstruction, which indicated the alignment in x and y directions was adequate.

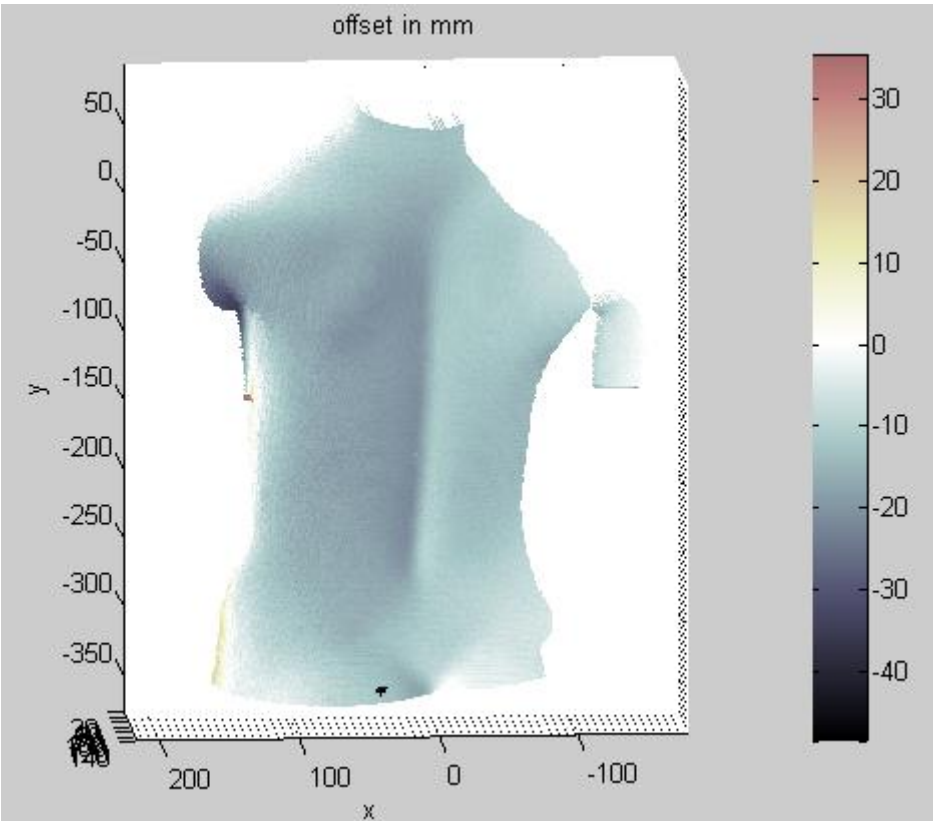


Figure 4.11. Rendered 3D surface from the structured light 3D reconstruction of the phantom seen in figure 4.9 (d).

The 3D reconstruction and visualization was repeated for several intentionally misplaced phantom locations. Note that in the absence of a room laser system, the reference point defined by the calibration board could not be accurately marked (after the removal of the calibration board, it became just a virtual point in space) and therefore the exact amount of misplacement from optimum could not be accurately measured. For this reason, measurements taken in this section were only intended as demonstrations of the visualization method. A common way of testing a vision system's capability in detecting misalignment was to introduce a known offset. The misplacement in these experiments, however, was introduced qualitatively based on the approximated location of the reference point.

The phantom was placed into the scene with a reasonably good *x*-*y* alignment to the optimal location, but at a *z*-location closer to the camera than it should be (for example, Figure 4.9 b). The visualization of this surface was in agreement with the introduced offset (see Figure 4.12). Compared to Figure 4.11, the entire surface darkened quite uniformly, showing the amount of *z*-offset had increased in the negative *z*-direction.

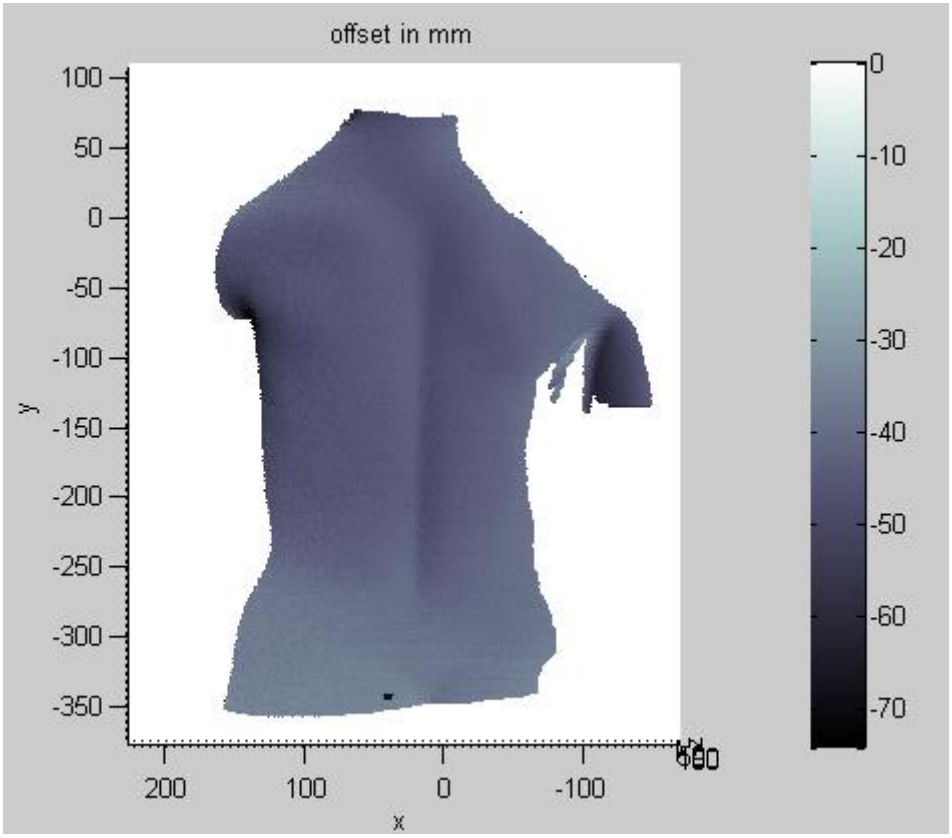
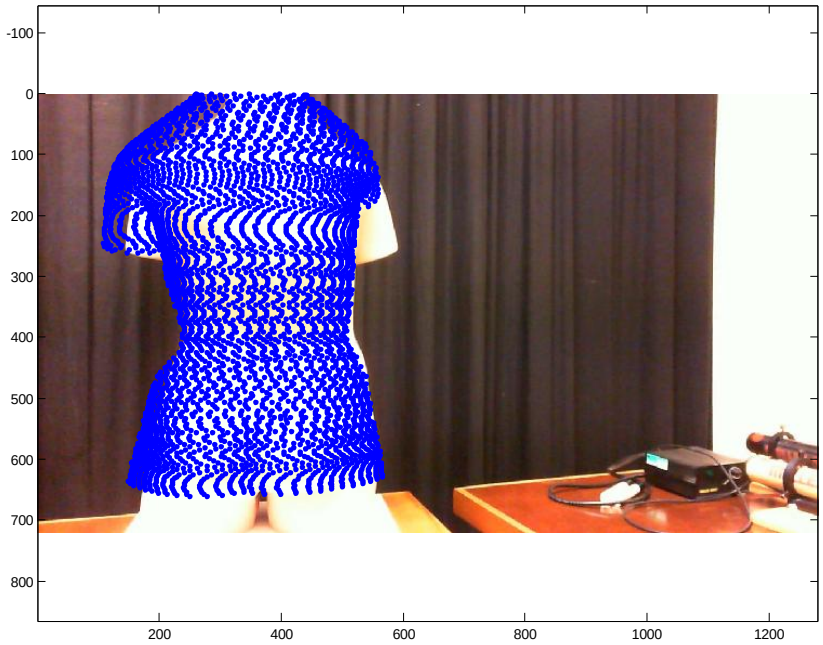
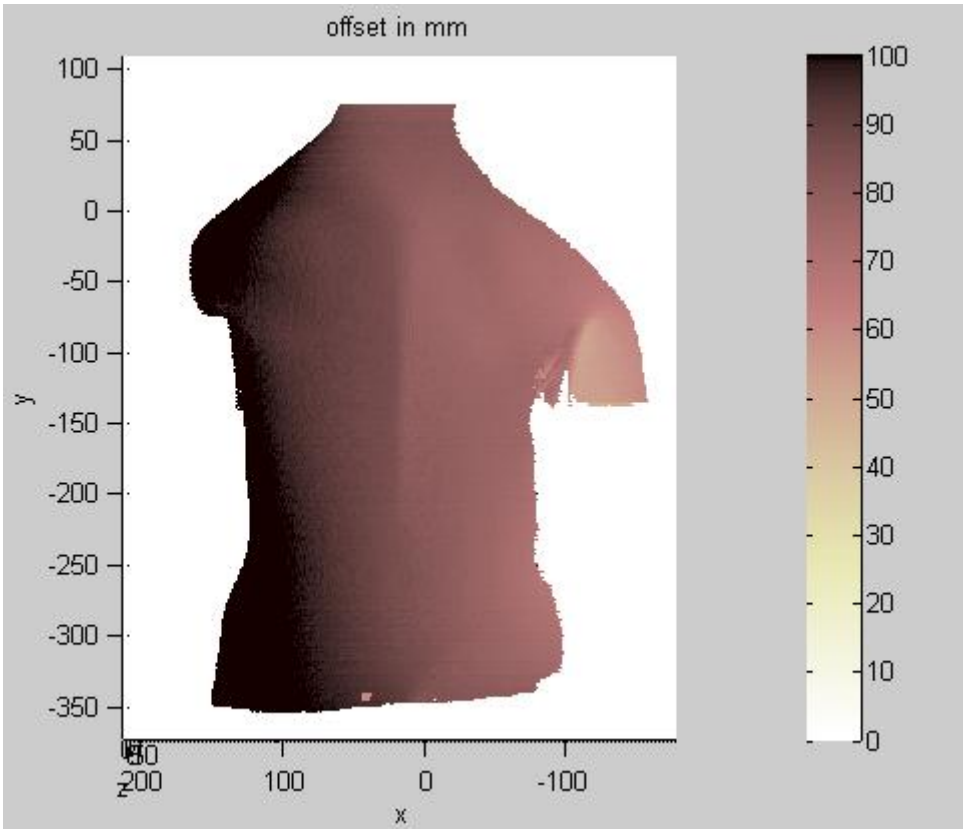


Figure 4.12. 3D surface of a phantom positioned closer to the camera than where it should be.

In the final example, the phantom was placed too far out in the *z*-direction, misaligned in *x* and *y* directions, and also had the wrong rotation (see Figure 4.13 a). All regions of the reconstruction had the “hot” colours showing the whole surface was further away from the optimal *z*-position. Unlike the previous two examples, the colour rendering across the surface varied strongly (see Figure 4.13 b), which indicates non-optimal alignment in dimensions other than the *z*-axis.



(a) AR showing a badly positioned phantom.



(b) Reconstruction of the phantom back.

Figure 4.13. AR alignment example, with large offsets in *x*, *y*, and *z* directions.

Chapter 5

Infrared 3D Vision

The Microsoft Kinect is a commercially available 3D camera designed for Microsoft's video gaming console Xbox. The hardware responsible for the 3D capabilities was designed by vision technology developer PrimeSense. The Kinect consists of a non-modulated infrared laser projector (therefore not capable of time-of-flight as some have originally speculated), an infrared sensor, and a 640-by-480-pixel camera. The details of how the Kinect reconstructs a 3D scene are not well documented as Microsoft has thus far not revealed the underlying algorithm. Infrared images of a Kinect in operation have shown that a large number of seemingly random dots is projected onto the scene. Apart from being invisible to the naked eye, the Kinect projection does not appear too dissimilar to the pseudo-random laser speckles used by commercially available, stereovision based products such as the AlignRT. Microsoft does not officially support the use of Kinect for any purpose other than on the Xbox, however, third party open source drivers for PC have been released on the internet. Amongst the third party drivers are the *OpenNI* framework and the accompanying *NITE* middleware [42], both released by PrimeSense itself, to open up access to the Kinect input signals via PCs.

The Kinect was setup with the aid of the HITLab (Human Interface Technology Laboratory), a campus based research institute dedicated to human-computer interaction, augmented reality, and recently a range of projects using the Kinect. A simple program written in C based on the PrimeSense drivers was provided by the HITLab. The program activates two panels showing a colour image and a depth map of the scene, both in real-time (see Figure 5.2). The user can hit the space key at any moment while the program is running to convert the depth map into a 3D point cloud and to save the point could data as a .txt file. The HITLab included the calibration parameters for the Kinect in the program, so no further calibration of the Kinect was necessary.

As described before, the infrared pattern projected by the camera seems to share some similarity with the laser speckles used to simplify the correspondence problem in stereovision. There are claims that Kinect is, in principle, a stereovision system,

with the key difference of one camera being infrared [30]. A simple test of this claim would be to cover up one of the camera at a time, because stereovision cannot function in the absence of one of the cameras. Figure 5.2 shows the result of such a test, the depth map of the scene was unaffected by the loss of the CMOS camera (Figure 5.2 bottom). This result suggested the underlying principle of the Kinect was perhaps more similar to that of a structured light system than a stereovision system.

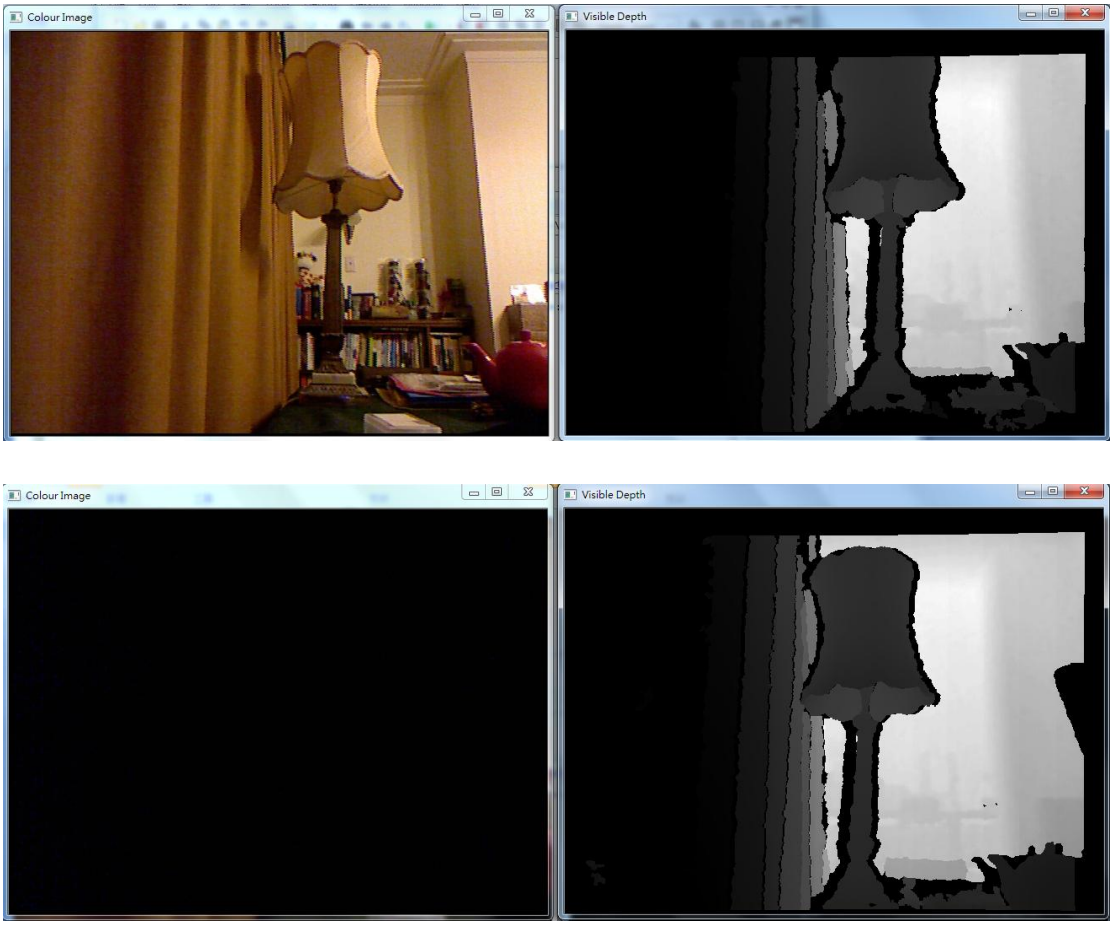


Figure 5.2. Kinect RGB image (left) and depth map (right) of a scene. The depth measurement was unaffected by the loss of the coloured camera (compare top row and bottom row).

3D surfaces of the anatomical phantom used in Chapter 4 were captured with the Kinect (see Figure 5.3, left). The Kinect was able to generate a depth map of the scene (for example, the right images of Figure 5.2 and Figure 5.3) in real-time. It was observed that the Kinect's depth registration tended to fail around the edges of objects, these regions show up on the depth map as dark patches. Regions where the effect was most prominent included the external borderlines in the camera perspective, of the lamp (see Figure 5.2), the phantom, and on the left edge of the board seen in the back ground (see Figure 5.3). The loss of depth information

resulted in missing points when the point cloud of the scene was reconstructed. This gave the edges of objects a "rugged" appearance (see Figure 5.4).

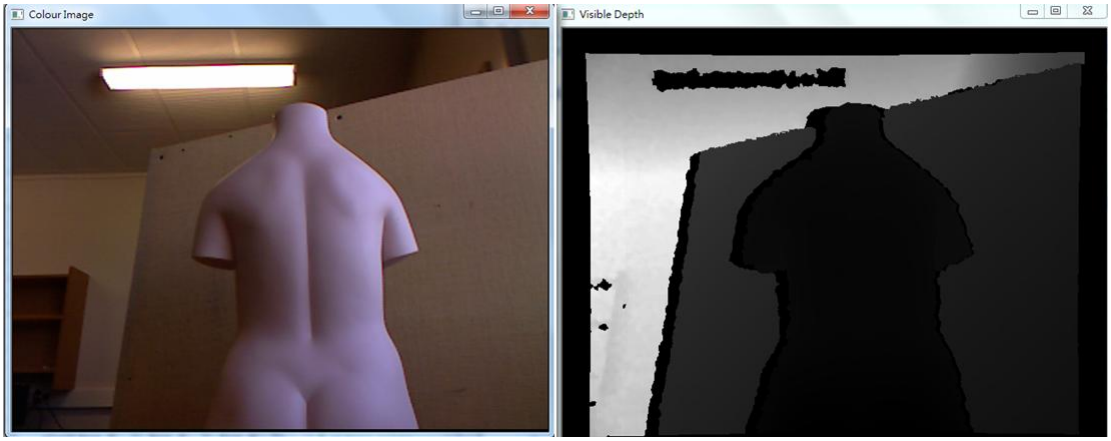


Figure 5.3. Capturing the anatomical phantom with the Kinect. Coloured image of the scene (left), and the depth map (right).

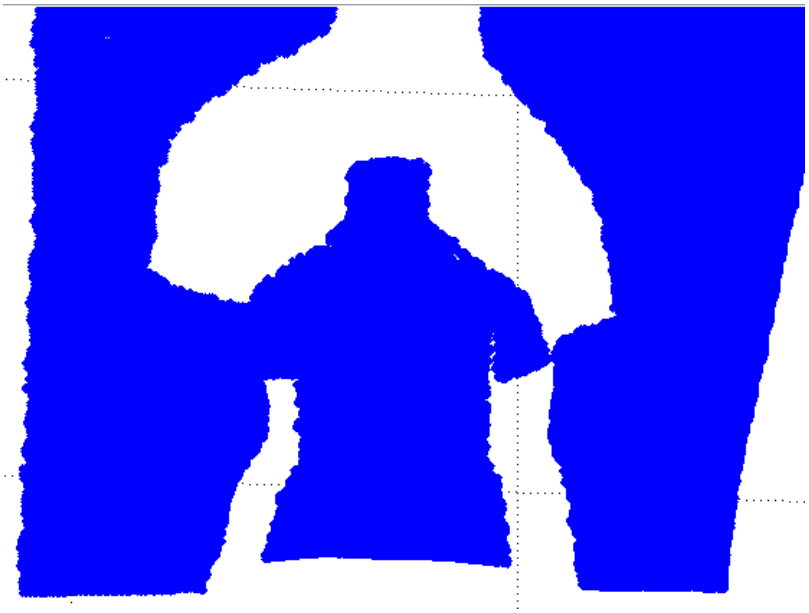


Figure 5.4. Reconstructed point cloud of the scene, zoomed in on the anatomical phantom.

Upon closer examination of the Kinect captured point cloud, points appeared to be in clusters of constant *z*, and variable *x* and *y* components (see Figure 5.5). The reconstruction of phantom and the surrounding scene consisted a total of 3.072×10^5 points, of these points, there were 57057 unique *x* values, 41534 unique *y* values, but only 528 unique *z* values which corresponds to depth. In this simple 3D scene acquisition test, it was found that while the Kinect offered real-time depth detection, the depth resolution was poor.

The observed limitation on depth resolution could potentially be compensated by an appropriate interpolation technique. Further work, such as a detailed distance versus depth resolution profile, is required to determine the suitability of Kinect or similar devices in 3D vision based radiotherapy patient setup.

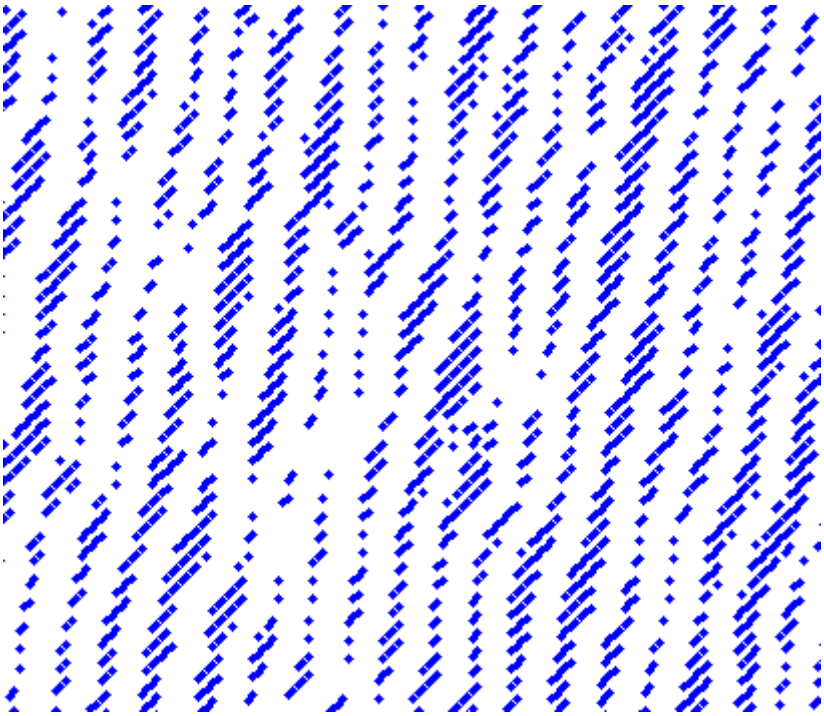


Figure 5.5. Close up view of the Kinect point cloud. Points appeared in clusters due to a lack of depth resolution.

Chapter 6

Discussion and Conclusion

6.1. Summary and Discussion

A 3D surface acquisition technique has been implemented to detect non-optimal posture and surface deformations in radiotherapy patient setup. The 3D technique was integrated with a simplified AR implementation to provide quantitative feedback on the accuracy achieved by AR guided qualitative alignment. While the existing AR system offered many useful features over the simplified version that was implemented (for example, it displays a rendered surface onto a live video stream instead of a point cloud overlaying a still image), the core of the existing AR system was programmed around the C++ based AR libraries OSGART, and ARToolKit. An AR program in Matlab was preferred due to unfamiliarity with the C++ language, as well as to keep the entire system in one programming language. Therefore a bare-bone version of AR was implemented from first principle in Matlab. This was sufficient for the purpose of demonstrating the viability of an AR 3D vision hybrid system. Future projects should consider working in the C++ language for the reason that a large number of functions, potentially useful for a 3D vision based patient alignment system, are already available in libraries such as OSGART, ARToolKit for AR, and OpenCV [43] for 3D computer vision.

In the full system implementation only the *z*-direction offset between the optimal patient outline and the acquired 3D patient contour were calculated. The justification for this was that the AR visual guidance provided an acceptable registration of the two surfaces. In future work, a quantitative method of registering the 3D reconstruction and the optimal surface outline should be implemented. A potentially useful tool for the programming of this feature is the Point Cloud Library (PCL) [44], an open sourced C++ library for point cloud processing. The PCL has a registration module that identifies corresponding points between two data sets and calculates the optimal transformation that minimizes the alignment error between corresponding points. This is in-effect a rigid body transformation calculation. However, it can potentially be used for posture adjustment, for example, if one arm is out of alignment, then a registration can be done on the part of the point cloud

representing that arm to provide information on how to correct it.
The accuracy of the structured light reconstruction was a constant obstacle throughout the project. Detection inaccuracies of ±1 mm, ±1.5 mm, and ±4 mm in the *x*, *y*, and *z*-directions respectively, were observed in a setup which was known to have zero offset (see section 3.2). Registration of the tracking board by extrinsic camera calibration was used to define the origin and coordinate system for AR. There are two main error sources in this procedure: the accuracy of fiducial marker detection and the accuracy of the intrinsic camera parameters estimated in camera calibration. In 3D structured light, a projector is used in conjunction with the camera. Therefore a third source of error, associated with projector calibration, is introduced. Projector calibration error was thought to be the main cause of the observed discrepancies between AR tracking and structured light reconstruction of the same board. In this project, efforts were made (such as increasing the mean triangulation angle) to reduce the effect of error introduced by the projector. However, the root cause of this was not fully investigated due to time constraints.

Temporal encoding is generally considered to only be suitable for stationary scenes. If the temporal encoding method is to be used on actual patients, it must be noted that any patient motions during the encoding phase will introduce additional errors not accounted for in this project. A camera and a projector capable of high frame rates are necessary, for example, if an encoding sequence the same length as the one used in this project is delivered at 30 frames per second, the encoding can be completed in less than 1.4 seconds. By using appropriate fixation devices and breathing techniques, it may be possible to obtain surfaces sufficiently stationary for the use of temporal encoding. However, the fact that spatial encoding requires only one image of the scene makes it fundamentally more suitable for dynamic scenes. In this project, the spatial encoding accuracy was undermined by the lack of camera resolution which leads to a lower spatial resolution and errors in colour recognition. In principle, with a more complete decoding algorithm the spatial encoding method should be capable of the same accuracy and resolution as the temporal encoding method.

6.2. Limitations of the Approach

One of the limitations of the structured light implementation was the strict requirements on room lightning. The spatial encoding technique, in particular, was highly susceptible to colour detection error. The implemented colour recognition was based on a set of user defined colour-channel ratios, which was highly sensitive to the background lighting as well as the intrinsic colouration of the object. The temporal encoding technique was more robust to changes in lighting condition. However, a dim background was desirable.

In a structured light 3D vision system, the position and orientation of the projector relative to the camera have to be held constant. A small shift of a few millimeters would warrant a re-calibration of the system.

It was observed that changing the lens settings on the projector would cause the vertical stripes on the encoding pattern to shift laterally. This effectively meant every projector calibration imposed constraints on the range of the 3D vision system. The projector could not properly encode objects outside the range constraints because the encoding pattern would be out of focus, and the projector lens could not be adjusted unless the projector was re-calibrated.

6.3. Further Work

A follow-on project utilizing the Microsoft Kinect for fast 3D surface acquisition and the PCL for point cloud registration is currently in progress.

Future work using a structured light computer vision method is not discouraged. However, personnel with an expertise in structured light computer vision and programming should be closely involved. Unlike stereovision and photogrammetry, structured light computer vision has received very little attention outside the computer vision community. The available literature on the practical aspects of the subject is mostly in proceedings of computer vision conferences, the experimental methods described often assume prior knowledge by the reader. The contents are therefore challenging for people from a different field. As an example, the use of combinatorial mathematics and dynamic programming is customary in the decoding of a spatially encoded scene. However, the details of their implementation are usually skipped.

An area potentially of interest for future projects is structured light using phase-shifting. Phase-shifting is a temporal encoding technique that produces a sine signal on a given object point over time. Unlike the encoding techniques implemented in this project, the achievable precision of the phase-shifting technique is no longer limited to integer projector coordinates (currently, the maximum number of points that can be uniquely encoded is the number of pixels available on the projector). This enables sub-pixel accuracy in projector coordinates. Sub-millimeter reconstruction accuracy has been reported using this technique [45].

6.4. Conclusions

A prototype alignment and position verification system for external beam radiotherapy patient setup has been developed. The system is intended to be used in addition to current setup procedures, by providing visual guidance as well as quantitative alignment accuracy without the use of ionizing radiation. The use of 3D surface acquisition also allowed surface deformations due to, for example, weight change, or non-optimal patient pose, to be quantified. The system utilized Augmented Reality for qualitative alignment (based on patient's CT scan) in the *x* and *y* directions, and a 3D computer vision technique using structured light to detect offsets along the camera's optical axis (*z*-axis). The accuracy achieved in this project was not sufficient to warrant clinical use. The measured error was mostly attributed to the 3D vision component of the overall system. Although it could not be shown in this project, the 3D structured light based methods have been demonstrated to be capable of sub-millimeter accuracy. One advantage of the concept presented in this project is that the guidance system can be setup with common office equipment, such as digital data projectors, and digital cameras, which most oncology department already have access to. The system is also highly customizable, in the sense that every component from hardware to the software can be modified to suit the particular requirements of each treatment, or upgraded when better components become available, and in most cases a calibration (section 2.3 and 2.4) is all that is necessary to incorporate the changes made.

References

1. Posgorsak (2005), *Radiation Oncology Physics*, IAEA 2005
2. Joiner, van der Kogel (2009), *Basic Clinical Radiobiology* 4th Edition, Hodder Arnold (London)
3. Verhey (1995), *Immobilizing and Positioning Patients for Radiotherapy*, Seminars in Radiation Oncology Vol. 5, 1995
4. Australian Radiation Incident Register (2006), *Summary of Radiation Incidents*, Australian Radiation Protection and Nuclear Safety Agency
5. Australian Radiation Incident Register (2007), *Summary of Radiation Incidents*, Australian Radiation Protection and Nuclear Safety Agency
6. UK National Patient Safety Agency (2007), *Quaterly Data Summary Issue 8*, National Health Service (NHS)
7. AAPM Task Group 104 (2009), *The Role of In-Room kV X-Ray Imaging for Patient Setup and Target Localization*, AAPM
8. Peng (2010), *Characterization of real-time surface image-guided stereotactic positioning system*, Med. Phys. 37, 5421
9. Fayad et al (2011), *Technical Note: Correlation of respiratory motion between external patient surface and internal anatomical landmarks*, Med. Phys. 38, 3157
10. Cervino et al (2010), *Frame-less and mask-less cranial stereotactic radiosurgery: a feasibility study*, Phys. Med. Biol. 55, 1863
11. Bert et al (2005), *A phantom evaluation of a stereo-vision surface imaging system for radiotherapy patient setup*, Med. Phys. 32, pp 2753-2762
12. Hughes et al (2009), *Assessment of two novel ventilatory surrogates for use in the delivery of gated/tracked radiotherapy for non-small cell lung cancer*, Radiotherapy and Oncology, 91, pp 336-341
13. Djajaoytra, Li (2005), *Real-time 3D surface-image-guided beam setup in radiotherapy of breast cancer*, Med. Phys. 32, pp 65-75
14. Rogus et al (1999), *Accuracy of a photogrammetry-based patient positioning and monitoring system for radiation therapy*, Med. Phys. 26, pp 721-728
15. Harris et al (2009), *Characterization of target volume changes during breast radiotherapy using implanted fiducial markers and portal imaging*, Int J Radiat Oncol Biol Phys 73(3):958-966
16. Chopra et al (2006), *Breast movement during normal and deep breathing, respiratory training and setup errors: implications for external beam partial*

breast irradiation, British Journal of Radiology 79, 766-773

17. Pathmanathan et al (2004), *Predicting tumour location by simulating large deformations of the breast using a 3D finite element model and nonlinear elasticity*, Lecture Notes in Computer Science 2004 - Springer
18. Armstrong (1998), *Target volume definition for three-dimensional conformal radiation therapy of lung cancer*, British Journal of Radiology 71, 587-594
19. Talbot (2009), *A Patient Position Guidance System in Radiotherapy Using Augmented Reality*, University of Canterbury MSc thesis
20. Meshlab, http://meshlab.sourceforge.net, website accessed on March 2011
21. VisionRT, AlignRT product overview, http://www.visionrt.com/page-154.html, website accessed on October 2011
22. Ogalem, Aloimonos (2005), *Shape and the Stereo Correspondence Problem*, International Journal of Computer Vision, 65(3), 147-162
23. Belhumeur (1992), *A Bayesian treatment of the stereo correspondence problem using half-occluded regions*, IEEE Computer Vision and Pattern Recognition (conference publication)
24. Cochran, Medioni (1992), *3D surface description from binocular stereo*, IEEE transactions on pattern analysis and machine intelligence, 14(10), 981-994
25. Lenman, Taubin (2009), *3D Photography for Beginners*, SIGGRPAH 2009 (conference publication)
26. Scharstein, Szeliski (2003), *High-Accuracy Stereo Depth Maps Using Structured Light*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003 (conference publication)
27. Salvi et al (2003), *Pattern codification strategies in structured light systems*, Pattern Recognition, 37, pp 827-849
28. Zhang (2002), *Rapid shape acquisition using colour structured light and multi-pass dynamic programming*, First International Symposium on 3D Data Processing Visualization and Transmission (conference publication)
29. Pages et al (2004), *A new optimised De Bruijn coding strategy for structured light patterns*, 17th International Conference on Pattern Recognition
30. Heikkila, Silven (1997), *A Four-step Camera Calibration Procedure with Implicit Image Correction*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1997 (conference publication)
31. Tsai (1987), *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal of Robotics and Automation, 3(4), 323-344
32. Haralick (1989), *Determining Camera Parameters from the Perspective Projection of a Rectangle*, Pattern Recognition, Vol. 22, No. 3, pp 225-230

33. Zhang (1999), *Flexible Camera Calibration By Viewing a Plane From Unknown Orientations*, 7th IEEE International Conference on Computer Vision 1997 (conference publication)
34. Bouguet (2010), Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/, website accessed on April 2011
35. Kleiner et al, Psychophysics Toolbox Version 3, http://psychtoolbox.org/HomePage, website accessed on April 2011
36. Kleiner et al (2007), What's new in Psychtoolbox-3, ECVP 2007 (conference presentation slideshow)
37. Brimijoin, O'Neill (2010), de Bruijn sequence generator for Matlab, http://www.mathworks.com/matlabcentral/fileexchange/28015-de-bruijn-sequence-generator, website accessed on May 2011
38. IPEM (1999), IPEM Report 81: *Physics Aspects of Quality Control in Radiotherapy*, Institute of Physics and Engineering in Medicine (International recommendation)
39. AAPM Task Group 142 (2009), *Quality Assurance of Medical Accelerators*, AAPM (International recommendation)
40. CERR, a computational environment for radiotherapy research, http://www.cerr.info/about.php, website accessed on May 2011
41. MathWorks (2009), Matlab (R2009b) product documentations, The MathWorks Inc
42. Fechteler et al (2007), *Fast and High Resolution 3D Face Scanning*, ICIP 2007 (conference publication)
43. OpenCV, http://opencv.willowgarage.com/wiki/, website accessed on August 2011
44. The Point Cloud Library, pointclouds.org, website accessed on March 2012
45. Sadlo et al (2005), *A Practical Structured Light Acquisition System for Point-Based Geometry and Texture*, Eurographics Symposium on Point-Based Graphics (conference publication)